Lecture Notes in Computer Science 5166

Andreas Butz Brian Fisher
Antonio Krüger Patrick Olivier
Marc Christie (Eds.)

Smart Graphics

9th International Symposium, SG 2008
Rennes, France, August 27-29, 2008
Proceedings

Springer

Volume Editors

Andreas Butz
Ludwig-Maximilians-Universität München
Institut für Informatik, LFE Medieninformatik
Amalienstrasse 17, 80333 München, Germany
E-mail: butz@ifi.lmu.de

Brian Fisher
Simon Fraser University at Surrey
13450 102 Avenue, Surrey BC V3T 5X3, Canada
E-mail: bfisher@sfu.ca

Antonio Krüger
Westfälische Wilhelms-Universität, Institute for Geoinformatics
Weselerstrasse 253, 48161 Münster, Germany
E-mail: antonio.krueger@wwu.de

Patrick Olivier
University of Newcastle upon Tyne, Informatics Research Institute
Newcastle upon Tyne, NE1 7RU, UK
E-mail: p.l.olivier@ncl.ac.uk

Marc Christie
Equipe BUNRAKU, Maître de Conférences - Détaché à l'INRIA Rennes
IRISA-INRIA Rennes, Campus de Beaulieu, 35042 Rennes Cedex, France
E-mail: marc.christie@irisa.fr

Library of Congress Control Number: 2008933457

CR Subject Classification (1998): I.3, I.2.10, H.5.2, I.3.7, I.4

LNCS Sublibrary: SL 6 – Image Processing, Computer Vision, Pattern Recognition, and Graphics

ISSN 0302-9743
ISBN-10 3-540-85410-X Springer Berlin Heidelberg New York
ISBN-13 978-3-540-85410-4 Springer Berlin Heidelberg New York

Springer is a part of Springer Science+Business Media

springer.com

Printed in Germany

Typesetting: Camera-ready by author, data conversion by Scientific Publishing Services, Chennai, India
Printed on acid-free paper SPIN: 12463794 06/3180 5 4 3 2 1 0

Preface

For centuries, artists and designers have been creating communicative graphics. With the advent of new forms of media, the emergence of paradigms such as ubiquitous computing, and the rapid evolution of interaction devices, there is a continuous cycle of renewal of the technologies and methods to support artists, interaction designers and developers.

Developing new approaches requires an understanding of the fundamentals of perception and cognition as they relate to interaction and communication technologies, together with artificial intelligence and computer graphics techniques to automate reasoning and enhance cognition. Smart Graphics is in essence an interdisciplinary endeavor and brings together the fields of computer graphics, artificial intelligence, cognitive science, graphic design and fine art.

The International Symposium on Smart Graphics 2008 was held on August 27–29 in Rennes, France. It was the ninth event in a series which originally started in 2000 as an American Association for Artificial Intelligence Spring Symposium and has taken place every year since then. Due to the high quality of the papers submitted this year, the Program Committee decided to accept 17 full papers (instead of the usual 15), 9 short papers and 3 system demonstrations. The acceptance rate for full papers was 34%.

This year's meeting included a discussion as to the nature of the shape, content and future of the event. Representatives from different communities were invited to give their opinions, and the organizing committee would like to warmly thank them here. Such questions as the ongoing viability of the symposium and the consequences of co-locating Smart Graphics with other larger research events led to interesting debates and have prepared the groundwork for what could be the future of the Smart Graphics conference series.

We would like to thank all authors and speakers for making this year's event such a success, the reviewers for their careful work, and the program committee for selecting and ordering the contributions for the final program. Special thanks go to the INRIA research institute and to the local organizers of the event (especially Edith Blin and Lena Baudoin) for taking care of all the financial and organizational aspects of the symposium.

August 2008

Andreas Butz
Marc Christie
Brian Fisher
Antonio Krüger
Patrick Olivier

Organization

Organization Committee

Andreas Butz	University of Munich, Germany
Marc Christie	INRIA Rennes, France
Brian Fisher	University of British Columbia, Canada
Antonio Krüger	University of Münster, Germany
Patrick Olivier	Newcastle University, UK

Program Committee

Elisabeth André	University of Augsburg
William Bares	Millsaps College
Marc Cavazza	Teeside University
Sarah Diamond	Ontario College of Art and Design
Stephane Donikian	INRIA Rennes
Steven Feiner	Columbia University
Veronique Gaildrat	IRIT, Paul Sabatier University Toulouse
Knut Hartmann	Flensburg University of Applied Science
Hiroshi Hosobe	National Institute of Informatics, Tokyo
Tsvi Kuflik	University of Haifa
Rainer Malaka	European Media Lab
Jun Mitani	University of Tsukuba
Shigeru Owada	Sony CSL
W. Bradford Paley	Digital Image Design
Bernhard Preim	University of Magdeburg
Thomas Rist	University of Applied Sciences, Augsburg
Shigeo Takahashi	University of Tokyo
Roberto Theron	University of Salamanca
Takafumi Saito	Tokyo University of Agriculture and Technology
Lucia Terrenghi	University of Munich
Massimo Zancanaro	ITC-irst Trento
Michelle Zhou	IBM T.J. Watson Research Center

Secondary Reviewers

Ragnar Bade	University of Magdeburg
Alexandra Baer	University of Magdeburg
Arno Krüger	University of Magdeburg
Konrad Mühler	University of Magdeburg
Christian Tietjen	University of Magdeburg

Supporting Institutions

The Smart Graphics Symposium 2008 was held in cooperation with Eurographics, AAAI, ACM Siggraph, ACM Siggart and ACM Sigchi. It has been supported by INRIA Rennes - Bretagne Atlantique, CNRS, LINA Laboratory of Nantes, University of Rennes, City of Rennes and Brittany territory.

Table of Contents

Sketching

Navigation and Selection

Studies and Evaluation

Camera Planning

Visualisation

Short Papers

Demos and Posters

Pillow: Interactive Flattening of a 3D Model for Plush Toy Design

Yuki Igarashi[1] and Takeo Igarashi[2]

[1] The University of Tokyo, 4-6-1 Komaba, Meguro, Tokyo, 153-8904, Japan
yukim@acm.org
http://www.den.rcast.u-tokyo.ac.jp/~yuki
[2] The University of Tokyo, 7-3-1 Hongo, Bunkyo, Tokyo, 113-0033, Japan

Abstract. Pillow is an interactive pattern-design system for creating plush toys. The user imports a 3D surface model into the system and specifies the segmentation boundaries interactively by drawing seam lines on the model surface. Based on this segmentation, the system generates a 2D pattern by flattening each outlined region, and then visualizes the shape of the resulting plush toy by applying a simple physics simulation. If the result is not satisfactory, the user can make different seam lines. This closed-loop framework allows users to experiment with various seam patterns before actually working on real fabric to obtain the best-looking result. The system also estimates total sewing time based on the total length of the seam lines. We implemented the system to produce plush toys and a balloon.

1 Introduction

To design an original plush toy, one needs to construct an appropriate 2D pattern which is very difficult. For this reason, all existing patterns have been developed by professional designers. To enable people to design their own original plush toys and similar products, we developed an interactive pattern-design system. Called Pillow, the system allows users to import a 3D model and interactively specify segmentation boundaries by drawing seam lines on the model surface. The system generates 2D patterns by flattening the user-defined segments and then simulates the shape of the plush toy. If the simulation is not satisfactory, the user can try another set of seam lines. This closed-loop interaction allows one to experiment with various seam line designs without actually sewing real fabric. The system makes it easier to construct 2D patterns, and allows users to design their own original plush toys and other products. The system also estimates sewing time based on the total length of seam lines. We produced some plush toys and a balloon using the system, and discuss the results below.

Mori and Igarashi [1] presented an interactive system that allows nonprofessional designers to develop their own original plush toys. The user interactively designs a 3D model by a sketching tool, and then the system generates a corresponding 2D pattern. However, the system is not useful when the user already has a 3D model. Julius et al. [2] presented an automatic segmentation algorithm

A. Butz et al. (Eds.): SG 2008, LNCS 5166, pp. 1–7, 2008.

for plush toys and demonstrated its viability by creating real toys using the machine-generated pattern. However, their method relies on purely geometric criteria, and it is difficult to capture the perceptually important features of the original shape. As a result, seams can appear in undesirable areas, such as across the face of a toy, or in ways that ignore the symmetry of the model. Our approach focuses on the user's artistic process of pattern design and provides tools to support the exploration process.

2 User Interface

Figure 2 shows a snapshot of the system. The left window shows the original 3D model, the center window shows the flattened 2D pattern, and the right window shows the reconstruction. Users import a 3D surface model (Fig. 1(a)), upon which they draw free-form lines to mark segmentation boundaries (Fig. 1(b)). These can be erased if necessary, Once all the segments are outlined, the system

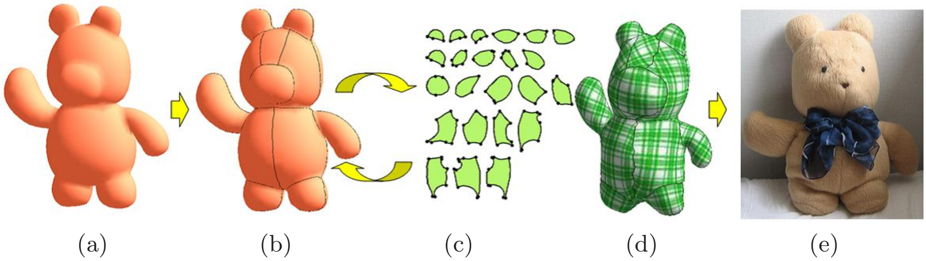


Fig. 1. Illustration of the Pillow system. (a) Input 3D model, (b) seam lines drawn by the user, (c) 2D sewing pattern generated by flattening the digitized regions, (d) reconstruction generated by applying a physics simulation to the pattern. The user repeats (b) through (d) until obtaining a satisfactory result. (e)The final product.

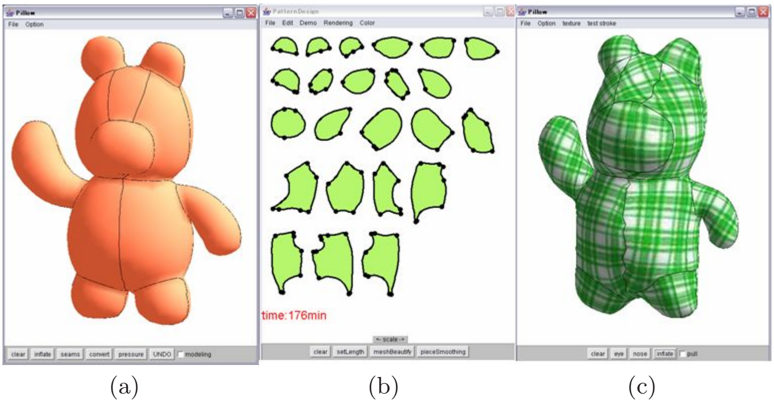
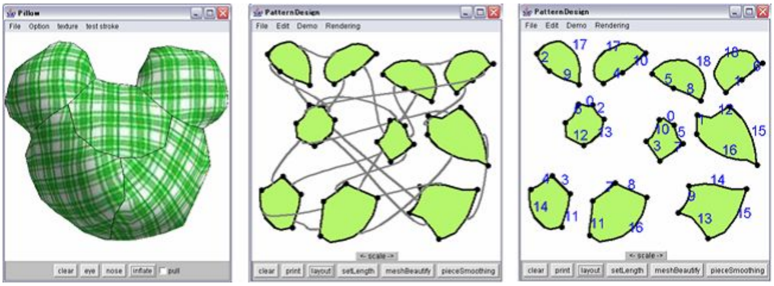

Pillow
File Option
clear inflate seams convert pressure UNDO □ modeling
PatternDesign
File Edit Demo Rendering Color
time:176min
<- scale ->
clear setLength meshBeautify pieceSmoothing
Pillow
File Option texture test stroke
clear eye nose inflate □ pull
(a)
(b)
(c)

Fig. 2. A screen snapshot of the Pillow system


Pillow
File Option texture test stroke
clear eye nose inflate □ pull
PatternDesign
File Edit Demo Rendering
<- scale ->
clear print layout setLength meshBeautify pieceSmoothing
PatternDesign
File Edit Demo Rendering
<- scale ->
clear print layout setLength meshBeautify pieceSmoothing

(a) (b) (c)

Fig. 3. Segments joined using connectors (b) and numbers (c)

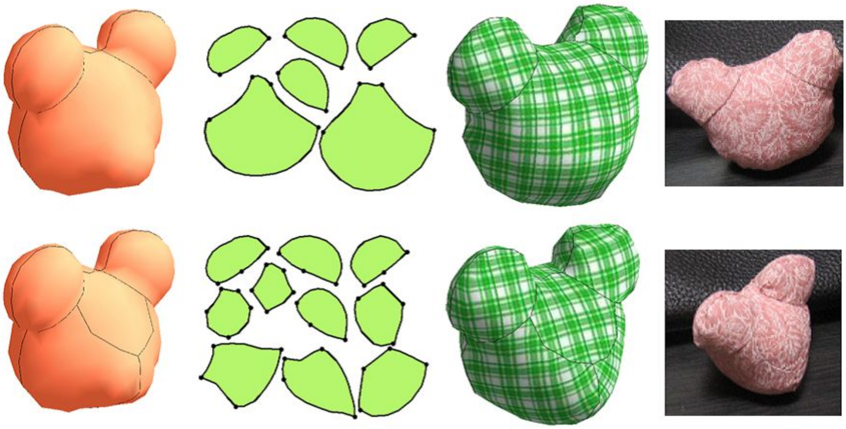


Fig. 4. Example of pattern designs

automatically flattens them (Fig. 1(c)) and reconstructs the 3D geometry by virtually sewing all of the flattened pieces together and applying a simple physics simulation (Fig. 1(d)). This view represents how the stuffed toy might look when a real toy is created using the current segmentation. If the user dislikes the result, he can quickly change the segmentation by erasing and redrawing the seam lines. The system displays how patches are joined by showing connectors or paired numbers (Fig. 3). Connectors are useful for understanding the relationship on the screen and numbers are useful as a printed reference on each patch. The system provides an automatic layout and manual arrangement interface for preparing the final pattern to be printed.

Figure 4 shows examples of pattern designs. Users can quickly experiment with various segmentation strategies using the system, and the simulation results successfully capture the overall shape of the final sewing results.

3 Estimation of Total Sewing Time

Total sewing time differs depending on the design as shown in Table 1, and it is very helpful to know how long it will take to create a plush toy during the design phase. To achieve this goal, our system automatically estimates total sewing time and presents this information to the user during seam design. It is assumed that the user will sew the pattern by hand.

Table 1. Geometric values of the 2D pattern

	Time(min)	L_{seam}	N_{seam}
Fig. 4(top)	40	2756.5	16
Fig. 4(bottom)	68	3706.3	38

3.1 Classification of Time Costs

Four main factors, or costs, determine the time required to create a plush toy: cutting a printed pattern (C_{cut}), tracing the pattern boundary on the cloth (C_{trace}), cutting the cloth (C_{cut_cloth}), and sewing the cloth (C_{sew}). The cost of stuffing is ignored because it is negligible compared to the sewing time. Therefore, the total cost (C_{total}) to create a real plush toy is defined as:

$$C_{total} = C_{cut} + C_{trace} + C_{cut\_cloth} + C_{sew}. \tag{1}$$

The cost of cutting a pattern (C_{cut}) has already been explored by Mitani [3]. We approximate C_{trace} and C_{cut_cloth} by using C_{cut} because all operations essentially trace the same seam lines on the 2D pattern. The total cost is written as follows:

$$C_{total} = 3C_{cut} + C_{sew}. \tag{2}$$

Cutting Cost: Mitani [3] computed the cost of cutting a pattern as follows:

$$C_{cut} = W_{cut}L_{seam} + W_{cut\_stoit}N_{cut\_stoit}, \tag{3}$$

where L_{seam} is the length of the seam line, W_{cut} is the weight per unit L_{seam}, N_{cut_stoit} is the number of stoit points where the scissors turn, and W_{cut_stoit} is the weight per stoit point. We use the weights $W_{cut} = 1.1s/cm$ and $W_{cut\_stoit} = 3.2s/stoit$, following [3]. The number of start and endpoints of the seams are N_{cut_stoit}, and therefore $N_{cut\_stoit} = N_{seam}$.

Sewing Cost: The cost of sewing (C_{sew}) increases proportionally to total seam length, which is the sum of the circumference of each piece. The cost of sewing all seam lines is represented as $W_{sew}L_{seam}$, where W_{sew} is the weight per unit L_{seam}. The cost of knotting at seam endpoints also affects sewing time. The cost of knotting is represented as $W_{knot}N_{knot}$, where N_{knot} is the total number of knots and W_{knot} is the cost per knot. N_{knot} equals $N_{seam}(= 2N_{seam}/2)$ because

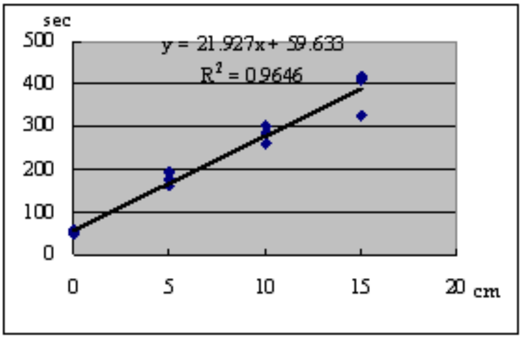


Fig. 5. Seam length and sewing time of a subject

the user knots at the start and endpoints of a seam. Therefore, the cost of sewing (C_{sew}) is represented as follows:

$$C_{sew} = W_{sew} L_{seam} + W_{knot} N_{knot}. \tag{4}$$

3.2 Estimation of Weights

We ran an experiment to estimate the two weights W_{sew} and W_{knot}. Five users sewed three lines (5, 10, and 15cm) by reverse-stitching, and we recorded their sewing times. According to the result, we set W_{sew} to 22s/cm and W_{knot} to 80s/knot.

3.3 Discussion

The experiment revealed that sewing time differs markedly between individuals. In addition, it is difficult to estimate the exact sewing time because of several unpredictable factors such as the physical interference between cloth and string. However, the study did reveal that for each individual, sewing time was well approximated as a linear function of the total seam length (correlation coefficients: 0.8-0.9 (Fig.5)). The prediction result provides valuable information to avoid excessively complicated seam designs.

4 Implementation

Pillow, our prototype system, is implemented as a Java$^{\text{TM}}$ program. Flattening and reconstruction run in real-time on a standard PC. We use ABF++ as the flattening algorithm [4]. It changes a 3D surface segment to a 2D pattern by minimizing the distortion in the angle space. For the physics simulation, we currently use a very simple mass-spring model. First, each vertex is moved in a normal direction to mimic outward pressure (Fig. 6(a)). Then, the system adjusts edge length (Fig. 6(b)). We plan to combine our system with more sophisticated simulation methods [5] in the future.

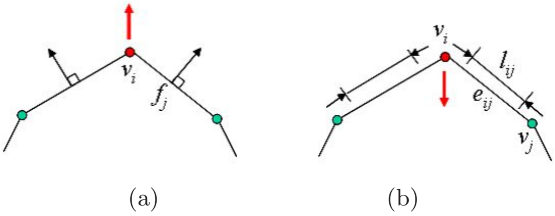


Fig. 6. Our simple model to mimic the effect of stuffing. (a) First, the system moves each face to its normal direction to simulate the effect of internal pressure; (b) then the system adjusts the length of each edge to preserve the integrity of the cloth.

5 Results

We created 2D patterns for a teddy bear model using our system and then created real plush toys using the resulting patterns. We used Teddy [6] for creating the input 3D model, and a standard polygonal mesh representation, so that users can use any polygonal model available on the Internet. It is also possible to create one's own model using a standard modeling program. Figure 1 shows an example of a seam design, simulation, and the final product. The simulation successfully captured the overall shape of the real plush toy. We also applied our method to design a balloon (Fig. 7). Figure 2 shows the pattern used to create this balloon. According to a professional balloon designer, the construction of a

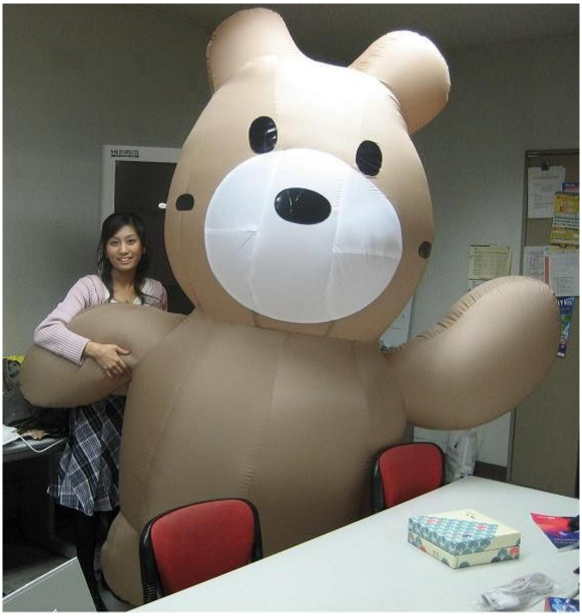


Fig. 7. A balloon created from the pattern shown in Figure 2

balloon can be roughly divided into five steps: creating a 3D model using standard 3D modeling software, manually designing a 2D pattern by cutting and pasting paper sections representing the 3D model (our model was a 1/10 scale version of the final balloon), scanning the 2D pattern and tracing the contours using a commercial drawing program, scaling the traced 2D patterns to actual size and printing it using a plotter, and cutting and sewing the 2D pattern. The required time from request to delivery is typically 1 month. Using Pillow, steps 1-3 can be significantly shortened (to about 2 weeks), thereby halving total production time.

6 Conclusion and Future Work

We developed Pillow, an interactive pattern-design system for creating plush toys and other products. Using the system, users can create 2D patterns easily, and experiment with various segmentations before actually sewing real fabric. In the future, we plan to improve the flattening algorithm to take cloth properties into account because existing algorithms (e.g., [7], [4]) for 2D parameterization do not consider cloth property. We also plan to incorporate more sophisticated simulation methods.

References

1. Mori, Y., Igarashi, T.: Plushie: An interactive design system for plush toys. ACM Transactions on Graphics (Proceedings of SIGGRAPH 2007) 23(3), Article No.45 (2007)
2. Julius, D., Kraevoy, V., Sheffer, A.: D-Charts: quasi developable mesh segmentation. Computer Graphics Forum (Proc. Eurographics 2004) 24(3), 981–990 (2004)
3. Mitani, J.: Research for a design support system for three-dimensional paper models using computer. The University of Tokyo Ph.D. dissertation (in Japanese) (2004)
4. Sheffer, A., Levy, B., Mogilnitsky, M., Bogomyakov, A.: ABF++: Fast and robust angle based flattening. ACM Transactions on Graphics 24(2), 311–330 (2005)
5. Grinspun, E., Krysl, P., Schroder, P.: CHARMS: A simple framework for adaptive simulation. ACM Transactions on Graphics 21(3), 281–290 (2002)
6. Igarashi, T., Matsuoka, S., Tanaka, H.: Teddy: A sketching interface for 3D freeform design. In: Proc. SIGGRAPH 1999, pp. 409–416 (1999)
7. Levy, B., Petitjean, S., Ray, N., Maillot, J.: Least squares conformal maps for automatic texture atlas generation. ACM Transactions on Graphics 21(3), 362–371 (2002)

Using the CAT for 3D Sketching in Front of Large Displays

Hongxin Zhang[1], Julien Hadim[2], and Xavier Granier[2]

[1] State key Lab of CAD&CG
Zhejiang University
Hangzhou 310027, P.R. China
zhx@cad.zju.edu.cn
[2] INRIA Bordeaux Sud-Ouest - Universities of Bordeaux,
351, cours de la Libération
33405 Talence, France
{hadim,granier}@labri.fr

Abstract. Sketching, as an intuitive tool for creation and edition of 3D prototypes, is a topic of increasing interest to the community. These approaches are based on the natural ability of humans to quickly draw in 2D some characteristic curves of 3D objects. Unfortunately, some 3D modeling operations - like positioning different components - and the modeling in front of large displays have still not reached the same level of intuitively in sketching systems. The main difficulty is to leverage the intuitive 2D gesture abilities of humans and lift them to XGCT sensical 3D operations. In this project, we present a new approach, based on a virtual 3D paper sheet metaphor and the use of a 6 degrees of freedom (DOF) device. With the associated interaction processes and visual feedbacks, it allows the user to quickly create and edit some sketched 3D models.

Keywords: Large Display, Sketch-based 3D Modeling, Interaction Devices.

1 Motivation

With the constant emergence and integration of new tools, current 3D modelers can become very complex to manipulate and can thus be very intimidating for non-expert users. Due to complicated operations and non-intuitive parameters, large training period are necessary for an user to be confident in 3D modeling. Creating 3D prototypes with mouse and keyboard only can thus become a slow and painful task. This problem has lead, during the recent years, to research and development of new approaches for simpler user interfaces. Some of the proposed solutions use the natural human ability to quickly sketch a global overview of an object. These approaches are naturally referred to as *3D Sketching*.

Research on 3D sketching has generated a new modeling approach: the system automatically infers a 3D shape according to a set of sketched 2D curves. Users can edit this model afterward to add details and to improve the geometric

A. Butz et al. (Eds.): SG 2008, LNCS 5166, pp. 8–19, 2008.

complexity with other advanced modeling operations (e.g., cutting and extrusion). All these operations are still based on sketching interactions. Since all 3D sketching systems use a drawing metaphor, they can be easily adopted even by non-experts.

Following the well-known sketching modeler “Teddy” [10], many solutions have been proposed for improving the surface generation [2, 6, 11, 18, 20] and the editing interactions [6, 16, 20]. Most of these previous systems are based on a classic 2D *pen-and-paper* context, similar to the use of a drawing tablet. In general (e.g., [2, 10, 20]), the final model is obtained by combining different components resulting from 2D sketches on different planes. However, combining these different components requires switching between 2D sketching interactions and 3D manipulations (mainly positioning and rotation). Hence in our project, we want to develop an integrated approach for these two modes, resulting in a more intuitive mode-switch.

The main aim in our project is to provide an intuitive manipulation environment for 3D design in virtual reality centers. In these environments, several users can collaborate and share ideas for creating 3D prototypes in front of a large display. In order to ease the interaction, we propose to work with an interaction device that can provide a direct mapping between its physical orientation/position and the relative orientation/position of a 3D object on the screen. This direct mapping guarantees that any position can be easily recovered by any user.

Our work introduces the following contributions. First, we show how 3D sketching can be formulated by using a 3D metaphor of virtual 3D paper sheet similar to the canvas introduced by Dorsey *et al.* [7]. Secondly, we introduce a full system, based on this metaphor and the use of a device with six degrees of freedom (DOFs). Finally, we present the proposed modeling steps and interactions, combined with some visual cues in order to provide a user-friendly solution.

This paper will be presented as follows. We first describe the existing solution for sketching in the context of very large screens and some existing solutions for 3D positioning. We then describe our reference solution, and the associated interaction steps required for modeling and editing 3D prototypes. Before concluding and introducing some future work, we illustrate this solution by detailed modeling sessions and by providing some results.

2 Previous Work

Among previous sketching solutions, implicit surfaces [1, 2, 11, 20] are the most commonly used surface representation since they provide a simple tool for creating complex objects, due to their natural blending between different components. This blending still requires correctly choosing their relative 3D positions.

One solution to the 3D positioning problem is to remove the restriction of sketching only 2D curves, by providing an interaction to directly draw 3D curves with a tracked device. Lapides et al. [12] use a tablet mounted on a support that can be translated vertically. One hand is used of the 2D curve drawing and the other one for vertical translation of the tablet. Unfortunately, a good 3D visualization system is required for accurate 3D drawing; otherwise, these approaches can result in undesired solutions due to the differences between the visualization system and our spatial representation of a 3D world. Furthermore, these techniques demand acute coordination abilities, involving both hands.

It is usually easier to have some reference planes [21] or surfaces [8] to draw on. Tsang et al. [21] use a set on planes oriented along the main axes. Markosian et al. [14] use the projection of the 3D curves on the image plane and its projection on a reference plane (called a "shadow curve") for inferring the 3D shape. The sequence of interactions is inverted by Grossman et al. [8] to extend the reference plane to a reference surface generated by the "shadow curve". They use a "Tape Drawing" [3] approach for curve sketching.

More recently, approaches based on multiple reference planes have been extended by Dorsey et al. [7] for architectural design and analysis. Their system uses strokes and planar "canvases" as basic primitives like for a traditional sketchbook (one sketch for each canvas) but can not provide true free-form models.

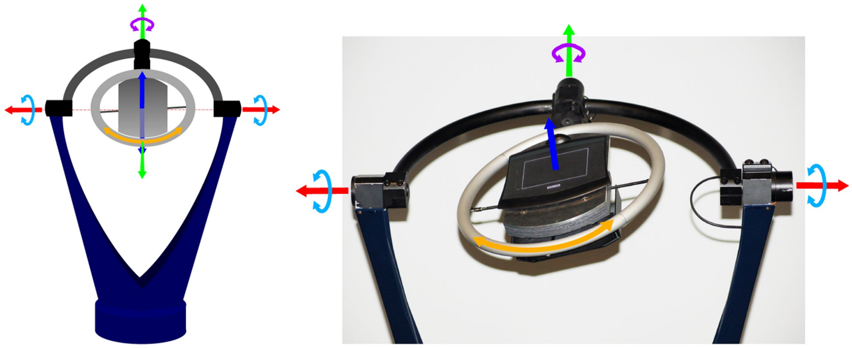
For an improved kinesthetic feedback, some tracked two-handed interactions have been introduced. Sachs et al.[17] use two Polemus-tracked devices: a palette as the reference plane and a stylus of the sketching part. Thanks to the palette, 3D positioning of the 2D curve or 3D object is easily achieved. Schroering et al. [19] use the same approach with camera-tracked devices: a board as reference plane and a laser pointer as a stylus. Unfortunately, as for any hand-held device, this can be physically tiring for the user. Furthermore, when multiple users are involved in the modeling process, the direct matching between the devices and the 3D positioning is lost anytime when the devices are released, put back at their original position, or passed to an other user. In our project, we want to reduce this limitation in the context of 3D sketching.

3 System Overview

We introduce a virtual 3D paper sheet metaphor in our sketching system. We assume that the user draws different sketches on different reference planes and combines them together in order to create the final object [7]. For the combination, we extend the classical 2D desk to a 3D space. On the modeling side, we use convolution surfaces [20]. These surfaces inherit the advantages of implicit surfaces, being expressed as a simple combination of the different components. On the hardware side, our system uses the CAT [9] to solve the relative 3D positioning of the different paper sheets.

3.1 Sketching Using Convolution Surfaces

In most sketching systems, everything begins with a 2D curve hand-drawn by users. Inheriting from the approach introduced by Teddy [10], a skeleton structure [1, 20] is extracted, based on the set of segments of the medial axis. A convolution with a linearly weighted kernel is then performed on each segment. By summing the fields of all segments, an analytical convolution surface is obtained. The resulting generic shape has a circular cross-section. New components can be



Synthetic view of the CAT

The real cat and its 6 DOFs

Fig. 1. Presenting the CAT and all 6 DOFs. The table can be rotated along all the 3 axes. Translations are detected by the user pressure along the 3 axes. Note the drawing tablet mounted on the table.

similarly designed by sketching on different projection planes, that we call *3D virtual paper sheet* for an interface point of view or *reference plane* for a modeling point of view. The convolution surface model smoothly combines the overlapping components. This approach overcomes several limitations of some sketched-based systems, including designing objects of arbitrary genus, objects with semi-sharp features, and the ability to easily generate a great variety of shapes.

On large displays, the quality of the resulting mesh is very important for displaying and for further manipulating the object. Hence regular topological connectivity and well sampled geometry are preferred. To achieve this goal, our current solution adopts the well-known polygonal tessellation – marching tetrahedra [4] – followed by a remeshing procedure [5] in order to smooth the final output.

3.2 Introduction to the CAT Interaction Device

Among the recent interaction devices, the CAT [9], a 6 DOFs interactor, provides us with a solution to the problems of 3D positioning. Compared to other devices, the CAT favors an unconstrained interaction since the user does not have to hold anything. Furthermore with the CAT, the rotations are directly controlled by an isotonic sensing mode while the (infinite) translations are controlled by an isometric sensing mode. Thanks to these features, it allows intuitive manipulations of 3D objects: the orientation of the table directly corresponds to the orientation of a plane in 3D space. Most similar devices like the SpaceMouse or 3D Mouse do not share such convenient properties. Our CAT-based solution can this simplify the orientation of the reference plane required for sketching. Furthermore, one one side, a translation is in theory an unbounded transformation and thus has to be relative. On the other side, a rotation is bounded and can thus be absolute. These interactions are naturally offered by the CAT.

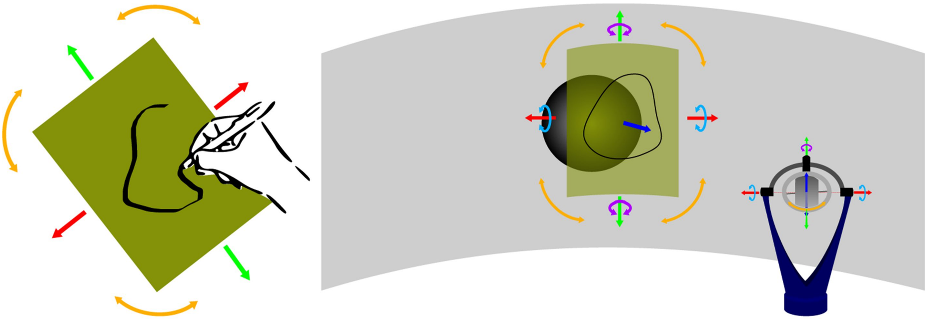
Moreover, a tablet fixed on the tabletop of the CAT is used for 2D interaction in our system. As required by most sketching solutions, a tablet can be directly mapped to a 3D virtual paper sheet, while keeping the user immersed in 3D environments (the 6 DOF of the CAT). Therefore, all the interaction sequences have thus to be carefully designed in order to use this 6 + 2 DOFs and will be described in the next section. With only one device, we can provide all the required DOFs.

3.3 The Virtual 3D Paper Sheet Metaphor

A sketching system is naturally based on a paper-and-pen metaphor. With the CAT, the tablet mounted on the top can be used as a reference plane, and thus directly associated to a 3D virtual paper sheet . The main problem for editing the different components of the object is thus to correctly position a set of reference planes in 3D (similar to the canvas in the work of Dorsey et al. [7]). Two main approaches can be used for this positioning: the traditional manipulation of the scene/object, or the manipulation of a virtual 3D paper sheet.

For the first one, user has to position the 3D scene/object relatively to a fixed 3D paper sheet. Unfortunately, it could be difficult to map the arbitrary shape of the resulting 3D object to the planar tabletop: the choice on the main orientation of the object corresponding to the orientation of the tabletop can be arbitrary.

On the CAT, the paper sheet can be directly associated with the tablet mounted on the moving table. We thus naturally decide to use the second approach: the positioning of the reference plane in 3D. The orientation of the table directly corresponds to the orientation of the 3D paper sheet (see Figure 2). The user just feels as if in front his desk, drawing multiple sketches on multiple paper sheets, and assembles them together. For the translations and rotations of the tablet are directly interpreted as the translation and the rotation of the virtual 3D paper sheet.



A paper sheet on a desk

Virtual 3D paper sheet on a large display

Fig. 2. Movement of a real paper sheet on a desk (left) and the corresponding mapping between the rotations and the translations of the CAT and the movements of the virtual 3D paper sheet (right).

4 Interactions for Object Creation

The modeling session is generally broken down in two main steps (see Figure 3). During the first one, the user is moving the supporting plane relatively to the 3D scene. Once this virtual 3D paper sheet correctly positioned, the user has to change to sketching mode. Thanks to the CAT, these two steps are performed using a single device. In order to ease the positioning of the paper sheet relatively to the scene, the virtual 3D paper sheet is transparent.

For the positioning mode, only the virtual plane is controlled by the user. The direct mapping between the table orientation and the virtual 3D paper sheet simplifies this process.

For the sketching mode, the relative positions of the virtual 3D paper sheet and of the 3D scene are linked together: any translation or rotation of the CAT is directly applied to the scene and the virtual 3D paper sheet. This has two main advantages. First, if the tablet is moving during the sketching session, the relative positioning is not lost. Second, for a user point of view, it is easier to draw with the table at a horizontal position. This is not always the configuration after the positioning of the virtual 3D paper sheet. To finalize the transition to

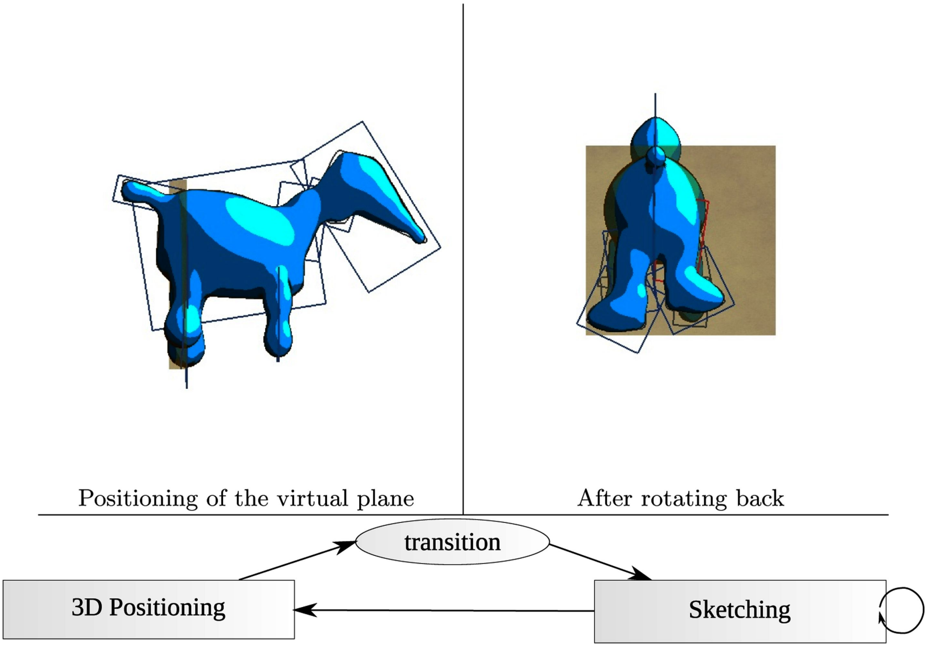


Fig. 3. Positioning of the virtual 3D paper sheet. The two modeling steps (upper images) and the associated interaction graph (lower image). During the 3D positioning mode, only the virtual plane is manipulated. During the sketching mode, the scene and the virtual 3D paper sheet are linked together. During the transition, users can rotate the virtual plane and the CAT to a horizontal configuration, more comfortable for drawing.

the sketching mode, she thus has to manually rotate back at this position and – since the object and the virtual plane are linked together – the whole scene (i.e., the 3D object and the virtual plane) is rotated in order to move back also the virtual 3D paper sheet in the initial position.

Note that, during the transition from the positioning mode to the sketching mode, the whole scene is also translated in order to move the center of the virtual 3D paper sheet at the center of the screen. This guarantees that, if the rotation back is applied to the horizontal position, the virtual 3D paper sheet will be back at the initial orientation and also centered on the screen, in a configuration similar to the original one (see upper-left image in Figure 3).

Once the silhouette sketched, we save its association with the reference plane. A reference plane is defined by a center $\boldsymbol{c}$ (the center of the silhouette) and a normal $\boldsymbol{n}$ (defined by the orientation of the virtual 3D paper sheet). We store also the corresponding 2D bounding box for display.

5 Interactions for Object Edition

Editing can be more complex. In the ideal case, when adding a new component to the final object, the previous 2-step approach is sufficient. To increase the freedom of the user, and the possibility of multi-users interaction, editing also needs to be supported. For most of the editing tasks, we need to retrieve the supporting plane corresponding to the component that we want to modify. This can be easily done once again using the CAT. The user moves the virtual plane in the 3D scene, and when it coarsely corresponds to the target plane, this one is selected. The user can thus return the table to the initial position (horizontal) and performs the 2D operations. Similarly to the modeling task, during this mode change, the whole scene is also rotated and translated in order to move back the virtual plane back to its original position.

The selection of the closest reference plane is done based on the plane orientation (i.e., normal $\boldsymbol{n}$) and its center $\boldsymbol{c}$. To this aim, we defined a weighted distance d_i, defined as

$$d_i\left(\boldsymbol{n},\boldsymbol{c}\right) = \left(1 - \left|\left\langle \boldsymbol{n}_i \cdot \boldsymbol{n} \right\rangle\right|\right) + \omega \left|\boldsymbol{c}_i - \boldsymbol{c}\right|,$$

where $\boldsymbol{n}_i$ (resp. $\boldsymbol{c}_i$) denotes the normal (resp. center) of the existing reference plane i. The weight ω can be adjusted depending on the scene's dimensions[1]. Note that we use the absolute value of the cosine between the two normals to detect the proximity between two orientations, and that the positions are normalized to the unit bounding box.

In order to assist the user in selecting the correct reference plane corresponding to the object's component he whishes to modify, we use color cues. We associate to each component a color, which is used to display the reference plane and its corresponding surface (see Figure 4). We select a set of perceptually different colors, exclugind red, as we use this color to highlight the current closest plane.

[1] We use $\omega = 0.1$ in our software, in order to give priority to the orientation.

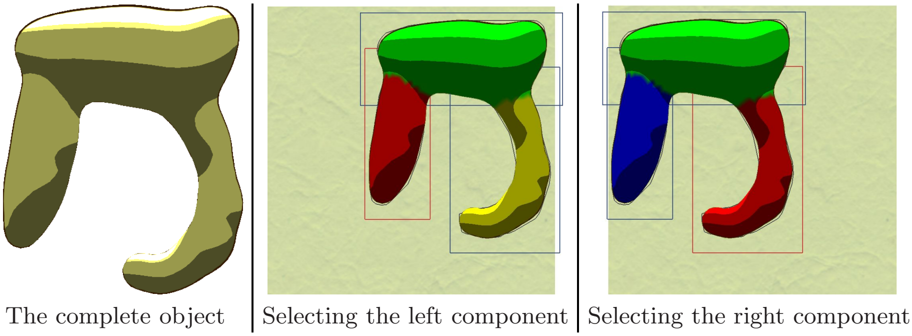


Fig. 4. Selection of a reference plane. In edit mode, each component has its own color and the selected component is highlighted in red. Note that we take the closest one to the center of the virtual 3D paper sheet.

6 Modeling Sessions

To illustrate a typical modeling session with the presented approach, we review the steps involved in creating the shape of the dolphin in Figure 5. The user first draws the main body of the dolphin using two components (steps 1 to 4). To rotate

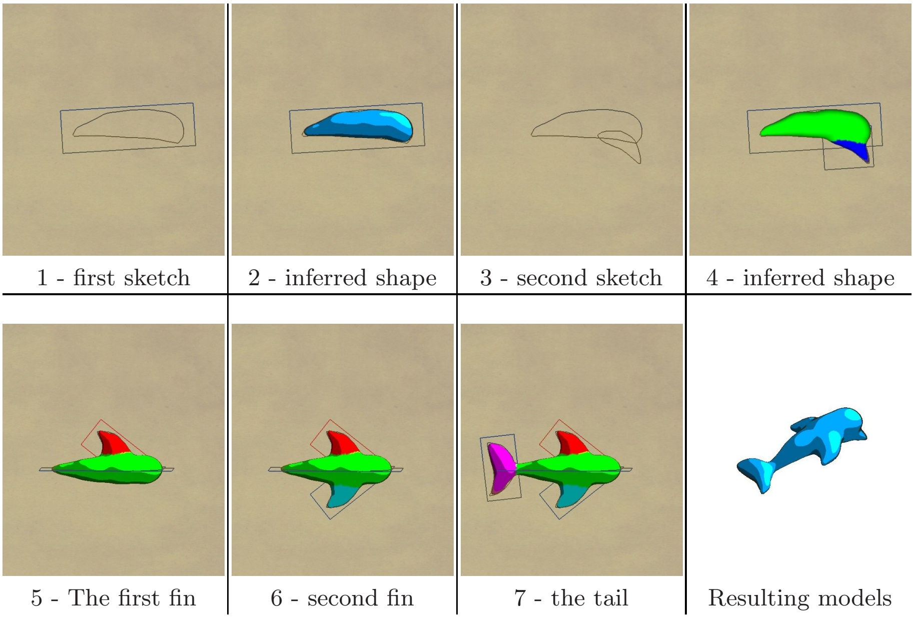


Fig. 5. Modeling a dolphin. The first four steps are done with nearly coplanar 3D paper sheet. During the transition from step 4 to step 5, the CAT has been used to position the virtual plane orthogonally to dolphin's body, and rotated back as explained in Section 4. The steps 5, 6 and 7 are now performed on the same plane.

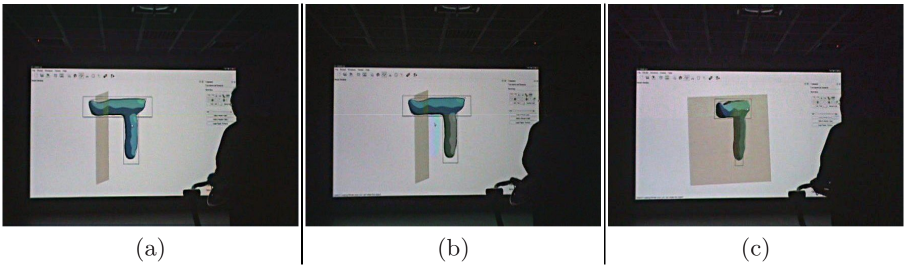


Fig. 6. 3D positioning of a new virtual plane. Once the position of the plane is validated by the user (a), a translation to the center is applied on both the plane and the object (b). The user can move back to horizontal position.

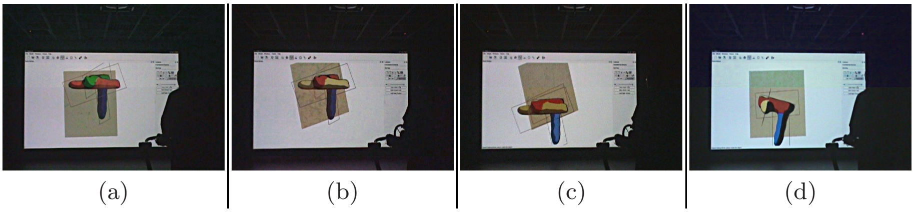


Fig. 7. Selecting an object component for editing. Beginning for the current position (a), the user moves the virtual paper sheet to select an other component (b). Each component has its own color, except for red which is used for selection. Once the selection is accepted by the user (c), a translation to the center is applied on both the plane and the object (d). The user can move back to horizontal position.

the plane at a position orthogonal to the main body, the CAT table top has to be vertical. Once back in sketching mode, the rotation of the CAT is applied to the virtual paper sheet and to the objects: the user can thus move back the virtual paper sheet to a nearly horizontal configuration, more comfortable for sketching.

This transition is also illustrated[2] in Figure 6. Not that, once the position of the virtual plane corresponds to the user's whish, the transition to sketching mode results in a translation of the whole scene (object and plane) to the center of the screen. This translation allows the user to quickly bring back the virtual sheet in front of him: the only action required is the rotation back of the CAT. During the whole process, the orientation of the CAT table always corresponds to the orientation of the virtual paper sheet. This sequence of actions is similar when it comes to select an existing component (cf. Figure 7).

7 Conclusion

In this paper, we presented new interactions for 3D sketching in front of large displays using the CAT device. The philosophy of our system is to provide a

[2] The images are samples of the associated video. The difference in color between the upper and lower part is due to our stereo system.

comfortable alternative 3D prototyping interface for large displays, with the intention of avoiding the inherent interaction drawbacks of traditional mouse-and-keyboard-based interfaces. The proposed system enables users to share modeling ideas freely by simply drawing 3D sketch curves and manipulating the CAT. We use a virtual 3D paper sheet as a natural modeling metaphor for 3D sketching. Such a 3D paper sheet can be conveniently used to help users determine orientation and depth information while sketching 3D curves. Moreover, this 3D paper sheet can be also applied to select, move, rotate, and modify specific curves so as to help users to achieve the prototyping goal. As demonstrated in the result section, even non-expert users can use our free-form modeler without any difficulty thanks to its ergonomic and comfortable interface. More user study has to be done in order to confirm this assumption.

Based on the presented approach, there are several potential directions and improvement can be explored in the future. First, due to the drawbacks of implicit surface representations, we can only prototype relatively rough and smooth objects. Working on improvement of skeleton extraction [1, 13] could be helpful to enhance the modeling capabilities ability of the system. Second, we will focus on the specific modeling applications to take advantages of field prior knowledge, such as sketch-based architectural design which has been already explored by [7]. Third, similar to the approach of Nealen et al. [15], the CAT can be applied to select a section of the object by positioning a 2D plane. Therefore, it could be useful in the geometrical operations of extrusion, cutting, sharpening or smoothing the resulting profile-curve [15]. Moreover, it is worth noting that deformations of reference plane are affordable solution to enlarge the realm of prototyping shapes.

Aknowledgement

This project is supported by the French embassy in China and the Egid. This work is also supported in part by the National Basic 973 Program of China (Grant No. 2002CB312102), the National 863 Program of China (Grant No. 2006AA01Z335), and the National Natural Science Foundation of China (Grant No. 60505001). We would like to thank Immersion S.A.S.[3] for the access to the CAT and to a Reality Center. We also want to greatly thank the anonymous reviewers for their comments and Jean-Sébastien Franco for proof-reading the final version.

References

1. Alexe, A.-I., Barthe, L., Gaildrat, V., Cani, M.-P.: A sketch-based modelling system using convolution surfaces. Technical Report IRIT-2005-17-R, IRIT, Université Paul Sabatier - Toulouse, France (July 2005)
2. De Araujo, B., Jorge, J.: Free-form modelling with variational implicit surfaces. In: Proc. of 12th Encontro Português de Computação Grafica (12th EPCG), October 2003, pp. 17–26 (2003)

[3] http://www.immersion.fr

3. Balakrishnan, R., Fitzmaurice, G., Kurtenbach, G., Buxton, W.: Digital tape drawing. In: UIST 1999: Proc. annual symposium on User interface software and technology, pp. 161–169. ACM, New York (1999)
4. Bloomenthal, J.: An implicit surface polygonizer. In: Graphics Gems IV, pp. 324–349. Academic Press, London (1994)
5. Botsch, M., Kobbelt, L.: A remeshing approach to multiresolution modeling. In: SGP 2004: Proc. Eurographics/ACM SIGGRAPH symposium on Geometry processing, pp. 185–192. ACM, New York (2004)
6. Cherlin, J.J., Samavati, F., Sousa, M.C., Jorge, J.A.: Sketch-based modeling with few strokes. In: SCCG 2005: Proc. spring conference on Computer graphics, pp. 137–145. ACM, New York (2005)
7. Dorsey, J., Xu, S., Smedresman, G., Rushmeier, H., McMillan, L.: The mental canvas: A tool for conceptual architectural design and analysis. In: Proc. Pacific Conference on Computer Graphics and Applications, October 2007, pp. 201–210. IEEE Computer Society, Los Alamitos (2007)
8. Grossman, T., Balakrishnan, R., Kurtenbach, G., Fitzmaurice, G., Khan, A., Buxton, B.: Creating principal 3D curves with digital tape drawing. In: CHI 2002: Proc. SIGCHI conference on Human factors in computing systems, pp. 121–128. ACM, New York (2002)
9. Hachet, M., Guitton, P.: The CAT - when mice are not enough. In: Proc. IEEE VR 2004 Workshop: Beyond Glove and Wand Based Interaction, March 2004, pp. 66–69 (2004) (Immersion S.A.S.), http://www.immersion.fr
10. Igarashi, T., Matsuoka, S., Tanaka, H.: Teddy: a sketching interface for 3D freeform design. In: SIGGRAPH 1999: Proc. annual conference on Computer graphics and interactive techniques, pp. 409–416. ACM/Addison-Wesley Publishing Co. (1999)
11. Karpenko, O., Hughes, J., Raskar, R.: Free-form Sketching with Variational Implicit Surfaces. Computer Graphics Forum (Proc. Annual Eurographics Conference 2002) 21(3), 585–594 (2002)
12. Lapides, P., Sharlin, E., Sousa, M.C., Streit, L.: The 3D tractus: A three-dimensional drawing board. In: IEEE International Workshop on Horizontal Interactive Human-Computer Systems (TableTop 2006) (January 2006)
13. Levet, F., Granier, X., Schlick, C.: Multi-view sketch-based freeform modeling. In: International Symposium on Smart Graphics (2007)
14. Markosian, L., Cohen, J.M., Crulli, T., Hughes, J.F.: Skin: A constructive approach to modeling free-form shapes. In: SIGGRAPH 1999: Proc. annual conference on Computer graphics and interactive techniques, August 1999, pp. 393–400. ACM, New York (1999)
15. Nealen, A., Igarashi, T., Sorkine, O., Alexa, M.: Fibermesh: designing freeform surfaces with 3D curves. ACM Trans. Graph. 26(3), 41 (2007)
16. Owada, S., Nielsen, F., Nakazawa, K., Igarashi, T.: A Sketching Interface for Modeling the Internal Structures of 3D Shapes. In: Butz, A., Krüger, A., Olivier, P. (eds.) SG 2003. LNCS, vol. 2733, pp. 49–57. Springer, Heidelberg (2003)
17. Sachs, E., Roberts, A., Stoops, D.: 3-Draw: A tool for designing 3D shapes. IEEE Comput. Graph. Appl. 11(6), 18–26 (1991)
18. Schmidt, R., Wyvill, B., Sousa, M.C., Jorge, J.A.: ShapeShop: Sketch-Based Solid Modeling with BlobTrees. In: Proc. Eurographics Workshop on Sketch-Based Interfaces and Modeling, pp. 53–62 (2005)

19. Schroering, M., Grimm, C., Pless, R.: A new input device for 3D sketching. Vision Interface, 311–318 (2003)
20. Tai, C.-L., Zhang, H., Fong, C.-K.: Prototype Modeling from Sketched Silhouettes based on Convolution Surfaces. Computer Graphics Forum 23(1), 71–83 (2004)
21. Tsang, S., Balakrishnan, R., Singh, K., Ranjan, A.: A suggestive interface for image guided 3D sketching. In: CHI 2004: Proc. SIGCHI conference on Human factors in computing systems, pp. 591–598. ACM, New York (2004)

MathPaper: Mathematical Sketching with Fluid Support for Interactive Computation

Robert Zeleznik[1], Timothy Miller[1],
Chuanjun Li[1], and Joseph J. LaViola Jr.[2]

[1] Brown University
[2] University of Central Florida

Abstract. We present MathPaper, a system for fluid pen-based entry and editing of mathematics with support for interactive computation. MathPaper provides a paper-like environment in which multiple mathematical expressions and even algorithms can be entered anywhere on the page. Mathematical expressions can also be modified using simple deletion and dragging gestures with real-time recognition and computation feedback. In addition, we support extended notations and gestures for controlling computational assistance, simplifying input, and entering algorithms, making MathPaper a user-friendly system for mathematical sketching and computation.

1 Introduction

Despite the tremendous computational power that is available on a typical desktop, it is interesting that students and professionals frequently resort to pencil and paper to do their homework or solve problems. Several factors conspire toward this phenomenon, such as the time it takes to start up an application and the limited display surface of a computer screen. However, we believe that the dominant reason for this reluctance to utilize desktop mathematics applications is the nature of pencil and paper itself, which affords fluid, direct specification of 2D mathematical notations and supporting diagrammatic structures in arbitrary juxtapositions.

From this motivation, we have developed MathPaper, a virtual equivalent to pencil and paper, that is enhanced to provide integrated computational support for mathematics. Ideally this should be at least as convenient as writing on paper but enhanced to recognize mathematical structures including multiple expressions, matrices, and even algorithms. Additionally, given the complexity of general mathematics recognition, including inherent notational ambiguities, we believe that interactive recognition feedback must be provided to ensure a fluid workflow. Previous systems [1,2] that require explicit interaction to display recognition feedback after an expression has been entered necessarily introduce a heavyweight cognitive step — users must shift from thinking about their problem at hand to examining the subtle symbolic and spatial aspects of an input expression to see if it matches their intentions. Instead, when provided with interactive recognition feedback, we believe users can amortize the inspection of

A. Butz et al. (Eds.): SG 2008, LNCS 5166, pp. 20–32, 2008.

recognition feedback over the course of their input. Changing to this lighter-weight process may make the essential nature of pencil-and-paper as a fluid, non-disruptive medium attainable.

Recognizing mathematical expressions non-disruptively, however, is only the first step in creating MathPaper. Additionally, we need extensions to standard mathematical notation to support computational assistance. Believing that mathematical notations are rich, familiar, and expressive, we attempted to introduce mathematical assistance in the form of enhanced mathematical notations. Thus, instead of treating computation as an external method to apply to a handwritten expression, we treat computation as an integral part of math notation. For example, drawing a $\Rightarrow$ to the right of any expression computes a numerical approximation of that expression. Although eventually some kind of computationally extended notation might be accepted as a standard that everyone would learn in school, we, in the meantime, need to provide additional support both for learning notational extensions and for accessing functionality that is more conveniently accessible through other user interface mechanisms.

In the next section of this paper, we discuss work related to the MathPaper prototype followed by a discussion of the fundamental interactions it supports. We then discuss how mathematics is specified and recognized in the MathPaper system and present some details on its support for computation. Finally, we present a brief discussion on an informal user evaluation along with future work and conclusions.

2 Related Work

Sketch-based interfaces have been explored for facilitating conceptual designs in various fields, including 2D diagrams [3] by Gross et al., mechanical design [4,5] by Alvarado and Kara, musical score creation [6] by Forsberg et al., 3D scenes [7] by Zeleznik et al., and freeform models [8] by Igarashi et al. In contrast, MathPaper provides a sketch-based interface for mathematical expression input, editing and computation.

For mathematical computation, sketch-based interfaces have also been developed by quite a few people and companies. Chan and Yeung developed a simple pen-based calculator PenCalc [9], and Thimbleby and Thimbleby also developed a calculator with a gesture-based interface supporting animation for simple math calculations [10]. MathBrush [11] recognizes mathematics and drives a symbolic algebra system with it. MathJournal$^{\text{TM}}$by xThink, Inc. can solve equations, perform symbolic computation, make graphs, and automatically segment multiple expressions, while Microsoft Math$^{\text{TM}}$ [1] supports both numeric and symbolic computation of a single expression. LaViola's MathPad^2 system [2] used a pen-based interface to let users create dynamic illustrations for exploring mathematical and physical concepts by combining handwritten mathematics and free-form drawings. Comparing with those sketch-based interfaces for mathematics, MathPaper provides a real-time, write-anywhere, fluid interface for mathematical computations. Expression recognition is automatic, with automatic segmentation for multiple expressions. Ink stroke deletion and dragging are supported with gestures, and matrices, including non-well-formed matrices, are supported together with algorithm sketches.

3 Fundamental Interactions

The fundamental interactions provided by MathPaper are for entering and editing mathematical expressions, issuing commands, and demonstrating the gestural user interface. With these interactions, our design philosophy was to ensure a high degree of real-time interactivity and feedback both at the recognition level and at the computation level (as in spreadsheets where changing different values automatically affect the results of various formulae).

3.1 Ink Input and Editing by Handwriting

Mathematical expressions are entered by writing virtual "ink", just as with paper. As Fig. 1 shows, multiple expressions are supported, and except for the cases where orders of expressions are important for computation, dependent expressions can be entered anywhere on the page, just like on paper. The right arrow is an extended notation for expression computation, detailed in Section 4.1.

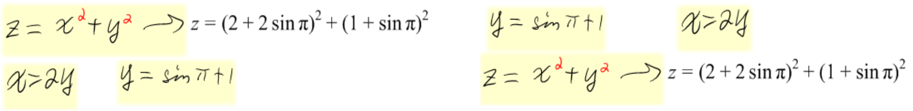

$z = x^2 + y^2 \rightarrow z = (2+2\sin\pi)^2 + (1+\sin\pi)^2$
$x = 2y$
$y = \sin\pi + 1$
$y = \sin\pi + 1$
$x = 2y$
$z = x^2 + y^2 \rightarrow z = (2+2\sin\pi)^2 + (1+\sin\pi)^2$

Fig. 1. Arrangements of dependent expressions. The expressions on the left are the same as on the right, yet arranged differently.

Editing of entered ink strokes can be done by circling and then dragging them or by scribbling over them to delete them; recognition of both gestures is discussed in Section 5.3. In addition, the horizontal bar of a square root or fraction can be extended by just drawing another horizontal line in the same direction with one end near the end of the line to be extended.

3.2 Commands

Since we provide more commands in our system than we can provide easy gestural or notational shortcuts for, we make use of widgets attached to each written expression to provide additional functionality. However, adding a normal widget would take up screen space, preventing the user from entering ink in that location. To conform to our write anywhere aesthetic, we avoid this problem by using a technique inspired by hover widgets [12]. We draw a green square under the ink on the display; hovering the pen tip over the square without touching the screen causes a cluster of buttons to appear to the upper-right (see Fig. 2). If

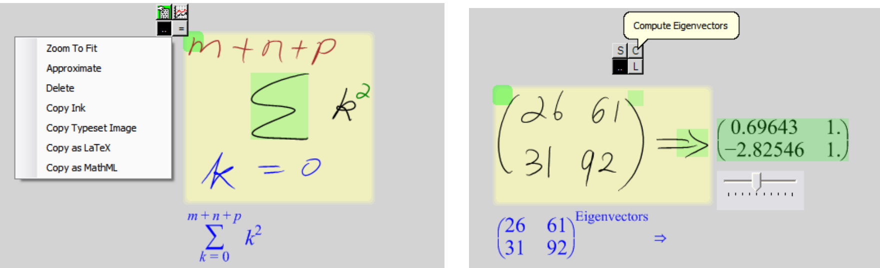


Fig. 2. Embedded widgets associated with specific expression locations

the pen moves directly towards the buttons, they can be pressed and interacted with as usual to display menus, perform functions, etc. If the pen moves in any other direction first, the buttons are removed. Thus, the user may draw strokes at any location on the screen and still have access to local widgets. If the user happens to move accidentally over the buttons, it is a simple matter to move back off again, although, in practice, this rarely occurs.

A small green square is provided at the top left of each expression to support various expression level commands such as graphing and simplification. For matrices, a green square is also placed at the top right with matrix-specific operations such as computation of eigenvectors, singular value decompositions, and norms. (The green boxes in figures other than Fig. 2 have been deliberately set to have the same color as the background to reduce distraction in this paper).

Demonstrations of extended notations and gestures. We attempt to make some of the extended notations and gestures we provide (see Section 4.1) discoverable by demonstrating to the user what the gesture would have been to perform a command the user has selected. For instance, if the user selects Delete from the menu, the system draws a scribble as an animation over the ink before deletion. Currently this happens for the Delete, Simplify, Numerically Approximate, and Graph commands. In the case of the notations for Simplify and Numerically Approximate, the ink for the notation is left on the screen as if the user had written it, since these are intended to be persistent marks.

3.3 Interaction for Switching between Drawing and Math Modes

In addition to mathematical expressions, MathPaper also supports drawing of freeform shapes and/or annotations without recognition or interpretation. Mode switching between math mode and drawing mode can be done either by choosing a command or by flicking a diagonal line followed by drawing a circle on the line. Ink in the freeform drawing mode is colored in blue. Annotations can also be later recognized as mathematical expressions; this is done by lassoing the ink strokes in the drawing mode followed by drawing a line entering the circle as shown in Fig. 3.

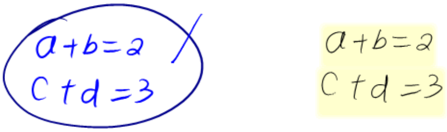


Fig. 3. Recognition of an annotation as math by lassoing the ink followed by an intersecting or crossing line stroke as shown on the left. The recognized strokes are shown on the right.

4 Specifying Mathematics

Our system currently supports entry of a solid range of basic math: a number of basic math symbols (including Greek letters and others such as ∞, $\mathbb{R}$, etc.), basic math relations ($<$, $\leq$, $\neq$, $\approx$, $\supset$, $\perp$, etc.), basic math operators ($+$, $-$, $\cdot$, $/$, $\wedge$, etc.), fractions written vertically ($\frac{a}{b}$), integrals, summations, roots, trigonometric and similar (log, ln, etc.) functions, and matrices. The recognized expressions can be exported as LaTeX code, MathML, and vector images of the typeset display.

Mathematics already uses duplicate notation for multiple purposes. For example, a textbook one author has uses u_x and u_y for the partial derivatives of a function u with respect to x and y, while other authors use that notation to represent the x and y coordinates of u. Perhaps the most common example is the use of each of i and j to represent either $\sqrt{-1}$ or an ordinary variable. Since, as a result, a mathematics recognition system already has to make choices when interpreting notation, it may as well consider recognizing other additions to standard notation to control further processing of the expressions, to take advantage of the pen's strengths to simplify entry, and for various other purposes.

4.1 Extended Mathematical Notations

The notational extensions discussed here extend users' control of their input, summarized in Table 1.

MathPaper supports interactive computation and graphing, both with arrow notations. A double right arrow at the end of an expression tells the system to compute a numerical approximation, while a single right arrow results in an exact symbolic evaluation, as shown on the left of Fig. 4. Since the only use we make of double arrows is for this numerical approximation function, an "arrow" referred to later in this paper is a single arrow unless otherwise specified.

Arrows are not only used for computation and graphing, but also for input brevity of matrices with patterns as shown on the right of Fig. 4. If a matrix has a certain pattern, input of the matrix can be simplified by extending a diagonal arrow for specifying the pattern. The arrow starts from the element on the first column and the first row, and points to the element on the last column and the last row. As shown on the right of Fig. 4, the dimensions of the matrix can be specified in the bounding box of the arrow, and the row index and column index variables can also be specified along with the dimensions. The number of rows and the row index variable are below the arrow, and the number of columns and

Table 1. Extended Notations

Functions	Notations	Descriptions
Computation	$\rightarrow, \Rightarrow$	Evaluates expressions
Graphing	$\leftarrow \rightarrow \uparrow \downarrow \nearrow \searrow \nwarrow \swarrow$	Graphs an expression
Brevity	$\searrow$ in matrices	Facilitates input of matrices with patterns
Space Management	$\mid - \neg$	Adds more horizontal or vertical spaces in matrices
Algorithmic Notations	$\leftarrow$	for function definition and return
	$\forall$	a shortcut for *for*
	$\in$	for argument type specification and loops
	$\sim$	for omitted data. Order is enforced.
	$=$	for assignment and equality
	$//$	for comments; can cover multiple lines
	$//T()$	produces a table of trace results
	$\nearrow \searrow \longrightarrow$	sets a trace point if in function definition

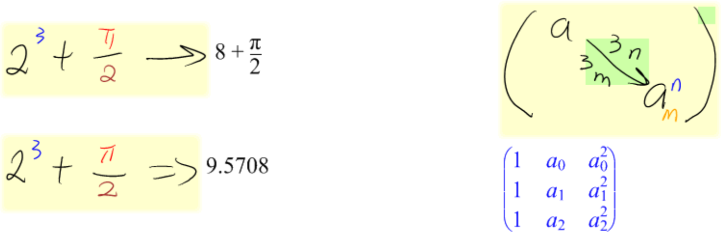

$2^3 + \frac{\pi}{2} \longrightarrow 8 + \frac{\pi}{2}$
$2^3 + \frac{\pi}{2} \Rightarrow 9.5708$
a
3
n
3
m
a_m^n
$\begin{pmatrix} 1 & a_0 & a_0^2 \\ 1 & a_1 & a_1^2 \\ 1 & a_2 & a_2^2 \end{pmatrix}$

Fig. 4. Examples of extended notations

the column variable are above the arrow. If elements above the main diagonal are a constant value, that value can be entered above and to the right of the arrow bounding box. Similarly, if elements below the main diagonal are constant, that value can be entered below and to the left of the arrow bounding box. This arrow extension allows for quick input of large matrices with certain patterns.

MathPaper also supports gestures for managing spaces between matrix elements. A vertical line that is at least twice the average height of the element bounding boxes moves the strokes on the right of the vertical line to the right a certain distance to allow insertions in the matrix. Similarly, a horizontal line that covers at least two matrix elements moves the strokes below by a certain distance. Since the long horizontal line can only add vertical space to more than one matrix element, we also use a ¬ gesture for adding more vertical space to any stroke below the horizontal part of the ¬. If any stroke is moved to the right, and the matrix has a closing parenthesis, the closing parenthesis is moved to the right accordingly. If a stroke intersects the space adding gesture, it is moved only if its center is to the right or below the gesture. For a multiple stroke symbol, movement of any stroke also moves all the other strokes of the symbol. These space adding gestures provide a faster way to move strokes than lassoing and dragging them if only horizontal or vertical space is needed.

Besides the above, MathPaper also supports extended notations for algorithm sketching [13] as listed in Table 1. Algorithm Sketching in MathPaper supports sketched algorithmic pseudocode with online computation capabilities. It supports flow of control constructs such as *for* loop, *if*, and *else*, and also supports a trace table for showing values of variables at any running point. A left arrow following function name with parameters specifies a function definition, while a heading left arrow returns function call results. For input simplicity, notations are extended for pen-based input. For example, $\forall$ is used for *for*, $\sim$ is used for ellipsis $(\cdots)$. Mathematical notation $\in$ is also supported for specifying argument types and ranges of iteration variables in *for* loops. The trace table is specified by the table head $//T()$, with trace variables listed in the parentheses. Right and diagonal arrows are extended for specifying the trace point in the algorithm, with the arrow starting from the trace point, and pointing to the trace table head. Comments can be added following $//$, and the extended notation $//$ can cover multiple lines, rather than only one line as used for keyboard and mouse input.

By extending notations, pen-based input can be simplified, and interactions can be more fluidly supported. All extended notations have specific meanings depending on context.

4.2 Allograph Mapping

MathPaper explicitly represents allographs, or different ways of writing the same symbol, with different user-specifiable interpretations for different allographs even when they are of the same symbol. As in our previous work [14], this means that, for instance, a script i could be mapped to the variable i while a straight-line i could be simultaneously mapped to $\sqrt{-1}$.

5 Recognizing Mathematics

Fig. 5 shows the major steps involved in the math recognition used in MathPaper. After recognition of an input symbol, affected ranges (each range represents one grouping of symbols that the system believes is a single complete expression) are re-parsed automatically. Symbols in one range are sorted and a baseline tree is constructed for the range. After a semantics parse step, an expression is generated for one affected range. More details about the recognition process are given in [13].

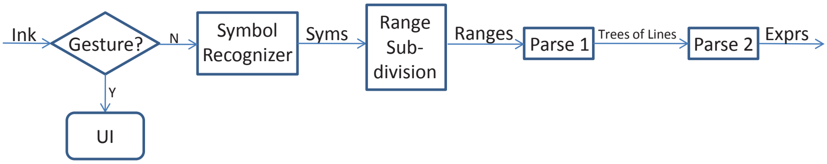


Fig. 5. Dataflow of mathematical expression recognition from ink strokes(from [13])

5.1 Interactive Feedback

MathPaper re-recognizes and re-parses after each update, including stroke entry or deletion, as well as dynamically in real-time during interactive movement. If a new stroke is added, it is recognized, and all the symbols in the affected range are sorted and parsed again with the immediate feedback as the typeset output. If more than one range is affected, for example by deleting a stroke which splits one range into two new ranges, or by adding a stroke which merges multiple ranges, all affected ranges are re-sorted and re-parsed.

Extended typeset conventions. The recognition results are generally displayed as 2D typeset expressions under the input strokes for each range, although there are alternate styles available [14]. Our typeset display algorithm has evolved to be very close to implementing the basic TEX display algorithm [15] with some additional knowledge about conventions that TEX relies on the user to follow. This feedback is only displayed under the expression the cursor was last over, to avoid confusion and clutter. One change we have made is to use a different typeface to distinguish the variable representing $\sqrt{-1}$ and the base of the natural logarithm each from the corresponding ordinary variables with the same symbols (i.e., i vs i and e vs e, respectively).

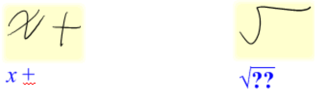


Fig. 6. Typeset convention extensions

In addition, we have explored extensions to indicate syntax errors by highlighting the relevant part of the output typeset with a red squiggle underneath it, as in Microsoft Word's indication of spelling mistakes. In the case of roots and vertical fractions we have found it more visually comprehensible to indicate a missing value by two bold question marks as shown in Fig. 6. In order to provide this feedback, our parser parses even mathematically invalid expressions, adding special nodes to the internal parse tree to indicate values that were assumed to be missing.

5.2 Automatic Range Segmentation

MathPaper supports multiple expressions, each represented by a range, and ranges are automatically segmented according to spaces between strokes, sizes and identities of the symbols at the boundaries of neighboring ranges.

After a stroke is recognized as a symbol, an inflated bounding box is computed for use in range identification. The amount to be inflated in each direction is based on the size, shape, and identity of the symbol. Special cases such as comma (,) and +/ =>< etc. are prevented from becoming isolated ranges even if they have small bounding boxes and relatively large distances to ranges on both sides.

After computing the inflated rectangle r for range identification, all existing ranges are examined against r using intersection tests. If an existing range's bounding box intersects r, the update belongs to the existing range. Otherwise, the existing range's bounding box is inflated. Let the inflated bounding box of the existing range be R. If R does not intersect r, the update does not belong to the existing range. If R intersects r, the bounding box of the nearest symbol in the range is inflated and if the inflated bounding box intersects r, the update hits the range. Otherwise the range is not hit by the update.

Hence, update is automatic not only when there is a single expression, but also when there are multiple expressions. Range segmentation is automatic after each update. After identifying all the hit ranges, symbols in them are re-sorted and parsed for the updated expressions.

5.3 Gesture Disambiguation

Gestures disambiguation is done in MathPaper by examining the contexts of the gestures. A stroke is recognized as a deletion gesture only if it intersects other strokes. As discussed above, lasso gestures are used to drag ink strokes around, and a stroke is recognized as a lasso only if it contains at least one stroke. Also for tapping as a gesture for the integral sign and summation sign upper limit and lower limit widget [14], it can be recognized as a tapping gesture only if it is in the bounding box of an integral or summation symbol. If a tapping is followed by another stroke which will make a symbol i or j, or a tapping can make a symbol i or j with a neighboring stroke, it is recognized as a symbol rather than a tapping gesture even if the tapping is inside the bounding box of an integral or summation symbol. Similarly, a single arrow can be used for symbolic computation, graphing, algorithm definition, etc, and its semantic meaning depends on its contexts. More detail on graphing gestures is found in Section 6.3.

6 Supporting Computation

Symbolic computation in our system is supported through mathematics engine plug-ins. Currently, only two are fully functional: one that uses Mathematica [16] as a back-end to do the computations, and the other, which supports a limited but still useful set of calculations, was developed in house.

6.1 Automatic Update

One of MathPaper's key features is its support of automatic updates for computations. If there is a change to an assignment expression, all computations are updated automatically as shown in Fig. 7 for the two computations of $a + b$ and $a^2 + b^2$ after the update of $c = T$ into $c = \pi$. Although the updated assignment might not be in a computation as shown for the update of $c = \pi$ in Fig. 7, this assignment update might affect other assignments, hence all computations need to be updated with any assignment update in order to have automatic computation updates.

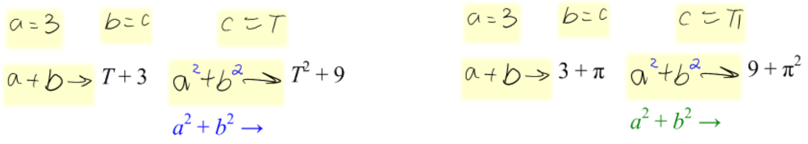


Fig. 7. Automatic update of computations after ink $c = T$ shown on the left is updated to $c = \pi$ shown on the right

6.2 Display Feedback and Interaction with Computation Results

If a computation result is asked for by the user, MathPaper attempts to determine if the result is a number. If the system is sure that the result is a number (whether exact or approximated), it displays the typeset feedback for the input expression in green rather than blue (see Fig. 7).

MathPaper also supports interaction with computation results if the results are numerical approximations. Numerical results are highlighted in green, and hovering over the results displays a sliding bar below the highlighted area as shown on the right of Fig. 2. The sliding bar is used to interactively change the desired number of decimal places that numerical results should be rounded to. An alternate option to the sliding bar is to use a horizontal stroke directly, or just hover over the results to specify the decimal places to round off.

6.3 Graphing

In addition to computation, MathPaper also supports graphing of expressions with an arrow gesture. If an arrow starts from within the bounding box of an expression, and points to outside of the bounding box of any expression, it is a graphing gesture. The graph will be plotted at the arrow head with the arrow ink being deleted. If a graphing arrow goes through multiple expression bounding boxes, all touched expressions will be plotted in one graph as shown in Fig. 8. MathPaper uses arrows rather than curved lines as used in MathPad^2 [2] for graphing to avoid a parenthesis being mis-recognized as a graphing gesture. To disambiguate a right arrow for symbolic evaluation from a graphing right arrow, a graphing right arrow must intersect at least one stroke in the expression, otherwise it will be interpreted as a computation gesture.

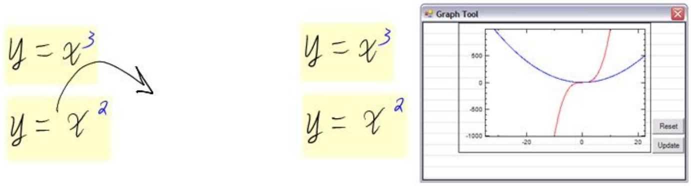


Fig. 8. Graphing of multiple expressions by one arrow

7 Discussion

We conducted an informal usability evaluation of MathPaper by providing the prototype to seven teachers of mathematics and science at both the high school and college levels. Along with the software, the teachers were also given an instruction manual and they were asked to experiment with MathPaper, focusing on entry, simplification, and graphing of simple mathematical expressions commonly found at the Algebra I level.

The majority of the teachers found MathPaper's interface fairly straightforward and easy to use and found the real-time recognition feedback useful in assisting them with correcting recognition mistakes. In addition, they found the real-time computation feedback, when modifying existing variables and expressions, to be an important part of MathPaper's functionality, since it let them quickly examine the results of any changes they made. They also reported that this feature is especially useful when teaching students how changes in different parts of a mathematical expression affect a computational result. The majority of the teachers found the graphing functionality to be somewhat lacking in terms of the types of graphs it supports but found its gestural interface straightforward. In total, the teachers enjoyed using MathPaper but also felt that it needed more mathematics computation functionality.

8 Future Work

MathPaper provides a fluid user interface for automatic recognition of multiple mathematical expressions and for mathematical computations. Currently, the user must explicitly switch modes between interpreted math and un-interpreted drawing. It would be more fluid to not have this drawing mode by relying on automatic detection of annotations or freeform drawing.

We have supported editing of ink strokes only, with no editing allowed for the typeset displayed. However, some MathPaper users have tended to try to edit the typeset directly. In the future, we would like to explore schemes for allowing editing of the typeset without taking available screen space away from starting new expressions.

Displaying of computation results has fixed locations and computation results cannot be reassigned to new variables. Being able to drag the results and to assign the results to new variables would be obviously quite helpful. Expression evaluation is supported, yet solving an equation for a specific variable is also important and should be supported in the future. Finally, a formal usability study is being developed for MathPaper that will explore how MathPaper can be accepted by users and how its UI can provide them with a more productive and enjoyable experience.

9 Conclusion

We have presented MathPaper, a mathematical sketching system for interactive computation. MathPaper supports automatic recognition of both mathematical expressions, including well-formed and non-well-formed matrices, and sketched algorithmic pseudocode, and automatic segmentation of multiple expressions. Expressions can be entered anywhere on the page, and ink strokes can be deleted, dragged and modified for error correction and expression re-arrangement. Mathematical notations have been extended and unambiguous gestures have been designed in MathPaper to support fluid interactions with mathematics.

Acknowledgements

This work is supported, in part, with grants from Microsoft Research and IARPA.

References

1. Microsoft: Math. Computer program, `http://www.microsoft.com/math`
2. LaViola, J., Zeleznik, R.: MathPaper^2: A system for the creation and exploration of mathematical sketches. ACM Transactions on Graphics 23(3), 432–440 (2004)
3. Gross, M.D., Do, E.Y.L.: Ambiguous intentions: a paper-like interface for creative design. In: UIST 1996: Proceedings of the 9th annual ACM symposium on User interface software and technology, pp. 183–192. ACM, New York (1996)
4. Alvarado, C.: A natural sketching environment: Bringing the computer into early stages of mechanical design. Technical report, Master's Thesis, Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology (May 2000)
5. Kara, L.B., Gennari, L., Stahovich, T.F.: A sketch-based interface for the design and analysis of simple vibratory mechanical systems. In: Proceedings of ASME International Design Engineering Technical Conferences (2004)
6. Forsberg, A., Dieterich, M., Zeleznik, R.: The music notepad. In: UIST 1998: Proceedings of the 11th annual ACM symposium on User interface software and technology, pp. 203–210. ACM, New York (1998)
7. Zeleznik, R.C., Herndon, K.P., Hughes, J.F.: SKETCH: an interface for sketching 3D scenes. In: SIGGRAPH 1996: Proceedings of the 23rd annual conference on Computer graphics and interactive techniques, pp. 163–170. ACM, New York (1996)
8. Igarashi, T., Matsuoka, S., Tanaka, H.: Teddy: a sketching interface for 3D freeform design. In: SIGGRAPH 1999: Proceedings of the 26th annual conference on Computer graphics and interactive techniques, pp. 409–416. ACM Press/Addison-Wesley Publishing Co., New York (1999)
9. Chan, K.F., Yeung, D.Y.: Pencalc: A novel application of on-line mathematical expression recognition technology. In: Proceedings of the Sixth International Conference on Document Analysis and Recognition, pp. 774–778 (September 2001)
10. Thimbleby, W., Thimbleby, H.: A novel gesture-based calculator and its design principles. In: MacKinnon, L., Bertelsen, O., Bryan-Kinns, N. (eds.) Proceedings 19th BCS HCI Conference, vol. 2, pp. 27–32. British Computer Society (2005)
11. Labahn, G., MacLean, S., Mirette, M., Rutherford, I., Tausky, D.: MathBrush: An experimental pen-based math system. In: Decker, W., Dewar, M., Kaltofen, E., Watt, S. (eds.) Challenges in Symbolic Computation Software. Dagstuhl Seminar Proceedings, Internationales Begegnungs- und Forschungszentrum fuer Informatik (IBFI), Schloss Dagstuhl, Germany, vol. 06271 (2006)

12. Grossman, T., Hinckley, K., Baudisch, P., Agrawala, M., Balakrishnan, R.: Hover widgets: Using the tracking state to extend the capabilities of pen-operated devices. In: Proceedings of CHI 2006, pp. 861–870 (April 2006)
13. Li, C., Miller, T., Zeleznik, R., LaViola, J.: AlgoSketch: Algorithm sketching and interactive computation. In: Proceedings of the 5th EUROGRAPHICS Workshop on Sketch-Based Interfaces and Modeling (SBIM 2008), pp. 175–182 (June 2008)
14. Zeleznik, R., Miller, T., Li, C.: Designing UI techniques for handwritten mathematics. In: Proceedings of the 4th EUROGRAPHICS Workshop on Sketch-Based Interfaces and Modeling (SBIM 2007) (August 2007)
15. Knuth, D.E.: The TEXbook. Addison Wesley, Reading (1986)
16. Wolfram, S.: Mathematica: A System for Doing Mathematics by Computer. Addison-Wesley, Reading (1988)

Intelligent Mouse-Based Object Group Selection

Hoda Dehmeshki and Wolfgang Stuerzlinger

Department of Computer Science and Engineering
York University, Toronto, Canada
hoda@cse.yorku.ca
www.cse.yorku.ca/∼wolfgang

Abstract. Modern graphical user interfaces support direct manipulation of objects and object groups. Current object group selection techniques such as lasso and rectangle selection can be time-consuming and error-prone. This paper presents a new approach to group selection that exploits the way human perception naturally groups objects, also known as Gestalt grouping. Based on known results from perception research, we present a novel method to group objects via models of the Gestalt principles of proximity and (curvi-)linearity. Then, we introduce several new mouse-based selection techniques that exploit these Gestalt groups. The results of a user study show that our new technique outperforms lasso and rectangle selection for object groups with an implicit structure, such as (curvi-)linear arrangements or clusters.

1 Introduction

Object group selection is an integral part of most graphical user interfaces. Most systems implement rectangle and/or lasso selection as well as shift-clicking. For the first two techniques the user drags a rectangle or a loop around the target items. Auto-complete lasso speeds up selection by connecting the end points of the loop automatically [9]. Shift-clicking involves clicking on each object in turn with the shift-key held down.

Although quite simple and powerful, rectangle and lasso selection are time consuming when the mouse-movement distance is large, e.g. when selecting large groups of objects or on large displays. Rectangle selection requires only traversal of the diagonal of the region and hence is often faster than lasso. However, it works only well for horizontally and vertically aligned arrangements. Shift-clicking is relatively time consuming for groups of targets with more three objects, but is the only alternative that can deal with randomly scattered targets.

In this paper, we present a new approach for group selection that addresses the shortcomings of current selection techniques for spatially contiguous target groups. Our approach is based on the way human perception naturally groups objects, also known as Gestalt grouping [6]. Gestalt grouping is normally explained via a set of principles. Proximity and good continuation are among the most important principles in the context of graphical user interfaces. Proximity states that *"being all other factors equal, the closer two elements are to each*

A. Butz et al. (Eds.): SG 2008, LNCS 5166, pp. 33–44, 2008.

other, the more likely they are to be perceived as belonging to the same form". Good continuation states that *"co-linear or nearly co-linear visual items tend to be grouped".*

2 Related Work

A substantial body of research in human perception has focused on measuring the strength of proximity and good continuity in dot patterns. Kubovy *et al.* [7] and Oeffelen *et al.* [13] model proximity as decreasing exponential functions of relative distances between dots. Feldman [3,4] introduced a model that describes the strength of collinearity among three or four dots as a function of inter-dot angles (angles between lines connecting successive dots).

Contour grouping has been studied extensively in the field of computer vision. Most relevant to the present discussion is work, which investigates perceptual grouping in the digital ink domain [1,10,12]. All these approaches are based on heuristic grouping functions, with the exception of [12]. Unfortunately, no usability evaluation of these approaches has been presented.

Spatial parsers employed in hypertext systems automatically recognize implicit structures (typically representing semantic relationships) among objects, e.g. [5,8]. These systems deal only with horizontal and vertical lists as well as clusters. They cannot deal with diagonal and curvilinear configurations. Moreover, they are typically based on heuristic functions that need to be tuned on a *per-user* basis. The only interaction technique offered is multi-clicking for hierarchical group selection. In ambiguous cases where there is more than one interpretation, most of the approaches visualize only the "best" interpretation. Neither interaction with secondary groups nor selection of multiple configurations is possible.

In the image editing domain, Saund *et al.* [11] applied a lattice-based grouping structure in which an object can be part of multiple groups. This system can detect curvilinear structures as well. Similar to spatial parsers, the only interaction technique is multiple-clicking for hierarchical group selection.

In our own previous work, we presented a system that recognizes implicit structures [2]. In the current paper, we improve on this work in multiple aspects. We utilize a perceptual-based scale-invariant proximity model and generalize the good continuity model to (curvi-)linear groups. Furthermore, we introduce several novel interaction techniques and present the results of a user study.

3 Motivation and Contributions

The approach we present in this paper is able to detect linear and curvilinear arrangements in arbitrary orientations as well as any form of clusters. Unlike previous work, our grouping methods are directly based on established models from perception science. This significantly increases the accuracy of Gestalt group detection and obviates the need to tune the system for individual users. Moreover, for ambiguous cases with more than one perceptual grouping, our system shows all interpretations at once. Based on different visual cues, the user can then not only distinguish the different configurations, but can also select the desired one(s). This enables selection of multiple groups at a larger scale without extra steps, for example. Finally, our approach extends existing interaction techniques in several ways. Most prominently, it enables partial group selection and allows the user to deal with ambiguous cases.

4 Grouping Objects by Gestalt Principles

Our system initially constructs a nearest neighbor graph. Then, it searches this graph to detect two types of perceptual groups: proximity and good continuity, where the last handles co-linearity as well as curvi-linearity.

4.1 Proximity Groups

Our proximity model is based on CODE, a scale-invariant dot-grouping algorithm [13]. In this algorithm, the grouping strength of each element (dot) exerted onto the others is modeled by a normal distribution function. This function is centered at each element and its standard deviation is half of the distance between the object and its closest neighbor. The strength of a proximity group is then defined as the summation over all individual functions. When the strength of a region surpasses a threshold, all elements in that region are grouped. Varying the threshold detects different groups at different scales. Figure 1 illustrates this on a group of four dots, labeled A, B, C, and D. Dashed and solid lines represent strength functions and the overall grouping strength, respectively. Applying threshold 1 puts all objects in the same group. Threshold 2 puts A, B, C in the same group and D in a separate group, etc.

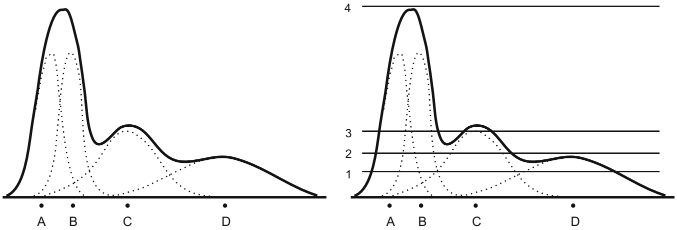


Fig. 1. Proximity grouping using CODE. Left: Objects are labeled A, B, C, and D. The dashed and solid curves represent the spread functions and the gradient strength, respectively. Right: Visualization of different thresholds.

4.2 Good Continuity Groups: Collinearity and Curvilinearity

For each set of four neighboring objects, our algorithm computes a linear coefficient (LC) indicating how strongly these four objects are perceived as a straight line. It is defined by:

$$\mathrm{LC} = \exp\left(-\frac{(a1+a2)^2}{2s^2(1+r)}\right) \times f(l_1, l_2, l_3).$$

where $a1$ and $a2$ are the angles between lines connecting the center of objects (see Fig. 2), r and s are constants, and f is a decaying exponential function of inter-object distances. This is based on Feldman's model for linear groupings of four consecutive dots [4]. We utilize a simpler model for groups of 3 objects [3].

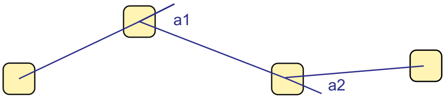


Fig. 2. Illustration of parameters used for good contiuity grouping

We extended these models to deal with arc groupings as follows: In the above equation, we substitute every inter-line angle α_i by $(\alpha_i - \alpha_{Avg})$ where α_{Avg} is the average of all line angles α_i's. Hence, a uniform curvilinear path gets a high grouping coefficient similar to the case of a straight path.

In an extra step, the initial collinear and curvilinear sets are repetitively merged to form longer groups. In each merging step, smaller groups are discarded only if the grouping coefficient is relatively weaker than a threshold.

5 User Interaction with Gestalt Group

Here we introduce several novel interaction techniques that allow single, multiple, or partial Gestalt group selection for spatially contiguous groups. Furthermore, we present a technique to resolve ambiguity.

Proximity Group Selection: The fundamental approach is similar to the multi-click approach used in text editors and spatial parsers. Clicking on an object in a cluster, i.e. a proximity group, selects the object itself. Double-clicking selects the cluster. Each successive click extends the selection with the closest cluster. For example, in Fig. 3 the first click on an object in cluster $C1$ selects the object. Double-clicking selects $C1$, the next click adds $C2$ to selection, etc.

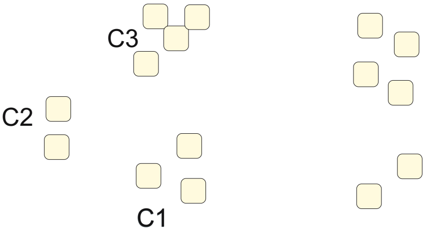


Fig. 3. Proximity group selection. Clicking on an object in cluster $C1$ selects the object. Double clicking selects $C1$, the next click adds cluster $C2$ to selection, etc.

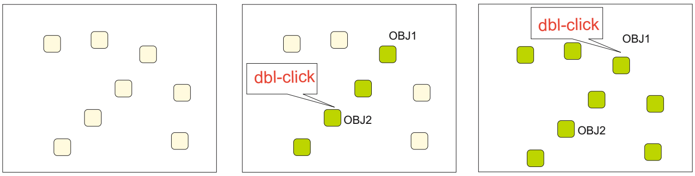


Fig. 4. Selecting good continuity group(s). Middle) Given the layout shown on the left, double clicking on $OBJ2$ selects the line. Right) Double clicking on $OBJ1$ selects the line and the arc.

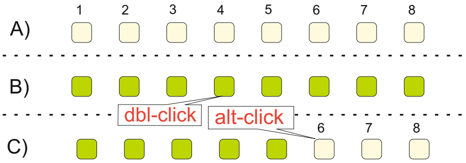


Fig. 5. Partial selection: A) a linear structure, B) double-clicking on object 4 selects the whole group, C) alt-clicking on object 6 then deselects objects 6, 7, and 8

Good Continuity Group Selection: Similar to cluster selection, clicking on an object in a good continuity group selects only the single object. Moreover, the good continuity group is also visualized as colored links between successive objects. Double clicking then selects the whole group. If the clicked object is part of multiple groups, all the groups are selected, see Fig. 4.

Partial Good Continuity Group Selection: To select a subgroup, the user first selects the whole group by double-clicking on an object, called anchor. Then the user deselects all undesired objects by clicking on the first non-desired one while holding down the alt-key. All objects on the path from the anchor "behind" this point will then be deselected, see Fig. 5.

5.1 Resolving Ambiguity

Ambiguity occurs when there is more than one visual interpretation of a scene. This is almost inevitable as soon as more than a few objects are involved. Our new approach permits the user to resolve ambiguities as follows:

– **Ambiguity in Clusters:** Grouping by proximity may not result in a unique grouping, as proximity can operate on multiple levels ranging from local to global. For example, in Fig. 3, three different configurations can be seen: five small groups, two large groups, or one whole group. Our grouping algorithm can detect all these configurations by changing the proximity threshold. Our novel interaction technique enables the user to change this threshold as follows: subsequent clicks on the same (anchor) object while holding the shift-key down changes the threshold to the next lower level, which enables group selection at larger scales. In contrast, subsequent clicks while holding the

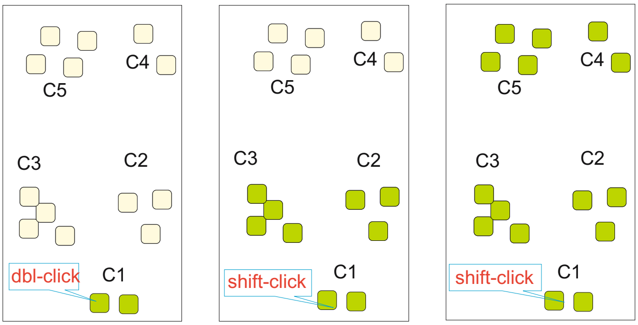


Fig. 6. Resolving proximity ambiguity: Left) Double-clicking on an object in $C1$ selects $C1$, i.e. the first level in the hierarchy. Middle) Then, a shift-click adds $C2$ and $C3$ to the selection (the next level). Right) Another shift-click selects all the clusters (the top level). At any level, an alt-click moves a level down in the hierarchy, i.e. one step left.

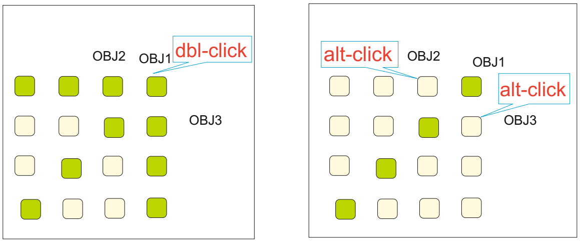


Fig. 7. Resolving curvilinear ambiguity: Left) double clicking on $OBJ1$ selects three groups (highlighted objects are selected). Right) clicking on $OBJ2$ and $OBJ3$ while holding the alt-key down deselect both groups.

alt-key down changes the threshold to the next higher level, which selects groups at a smaller scale, see Fig. 6.

- **Ambiguity in Curvilinear Groups:** As mentioned before, double-clicking on an object shared by multiple groups selects all the groups. If only one of them is desired, the rest can be deselected by alt-clicking on non-desired node(s), as with partial selection. Figure. 7 illustrates such a scenario. The user double clicks on $OBJ1$ to select the diagonal group. However, horizontal and vertical groups are also selected as they share $OBJ1$. Clicking on $OBJ2$ and $OBJ3$ while holding the alt-key down deselects them.

6 Experiments

We conducted a within subject study to assess the efficiency of our technique in comparison to rectangle selection and auto-complete lasso. For rectangle and lasso selection shift-clicking a single object adds it to the group. Ctrl-clicking toggles selection, similar to most current GUI applications.

To ensure a fair comparison, we designed the layouts so that the bounding box of all targets contained no distracters. The issue here is that close-by distracters affect each selection technique in a different way and hence distracters would act as a confounding factor (a fact confirmed by pilot studies). For example, for non-axis aligned configurations, distracters make rectangle selection much more difficult. For lasso selection, distracters make the "tunnel" the user has to traverse smaller, which usually results in a reduction in movement speed. Similarly, for perceptual based techniques, very dense configurations can contain many different perceptual groups, which force the user to choose among those groups. Hence, we designed the layouts so that for all three techniques no subsequent modification to the selected group was required. We designed 36 layouts for groups of square targets with different structures, classified as follows:

– 3 Arrangements: linear, arc, cluster
– 3 Sizes: small, medium, large
– 4 Orientations: horiz, vert, sq45, sq135

In linear and arc arrangements targets were placed along straight lines and arcs, respectively. In cluster arrangements, targets were randomly spread out over an area. Size was defined by the diameter of the bounding box of the target group (small ≈ 250, medium ≈ 450, and large ≈ 750 pixels). The horizontal and vertical orientations had a bounding box with a significant difference in height vs. length, at least 4 to 1 ratio; the diagonal arrangements had (almost) square bounding boxes with objects arranged at roughly 45 or 135 degrees.

6.1 Tasks and Stimuli

In each task, participants were asked to select targets within the above-mentioned layouts using auto-complete lasso, rectangle, or the new Gestalt-based technique. Targets and distracters were displayed in green or black, respectively. When an object was selected, its border became stippled and its color was desaturated. Moreover, correctly selected target objects changed their color to yellow for better discrimination. If only the correct targets were selected, a brief sound was played and the software advanced to the next task, with a different layout. Selection time was measured from the first mouse click after the layout was displayed to the time when only the correct targets were selected. The number of cancelations, i.e. when the user clicked on an empty area to de-select everything, or drew a new rectangle/lasso, was also recorded.

6.2 Experimental Design

We used a repeated measure within subject design. The independent variables were: Selection Technique (Gestalt-based, auto-complete lasso, or rectangle), Group Arrangement (linear, arc, or cluster), Size (small, medium, or large), and Orientation (horiz, vert, sq45, or sq135). Dependant variables were selection time and error rate. We counterbalanced the order of selection techniques with a 3x3 Latin square to compensate for potential learning effects. Participants were trained between 10 to 20 minutes on all three techniques by asking them to perform various selections on 12 practice layouts. The main experiment used 36 layouts (|shape|*|size|*|orientation| $= 3 * 3 * 4$), which were shown 3 times for each technique. This whole sequence was repeated 3 times in different orders. Hence, each participant performed a total of $36 * 3 * 3 = 324$ selections during the experiment. In summary, the experiment design was as follows:

- 12 training layouts
- 36 trial layouts, categorized by target structure:
  - shape (line, arc, cluster)
  - size (small, medium,large)
  - orientation (horz, vert, sq45, sq135)
- 3 selection techniques (lasso, rectangle, Gestalt)
- 3 repetitions
- 11 participants

A grand total of 3564 selections was performed in the experiment. At the end, each participant was asked to fill out a questionnaire to evaluate ease of use and learn-ability of our technique.

Apparatus: The experiments were conducted on a computer with a Pentium M 1.6 Ghz processor and 1 GB memory. Screen resolution was 1024x768 and an optical mouse was used. The software was written in Python.

Participants: Eleven students from a local university campus were recruited to participate in the experiment: six females and five males, between 25-35 years of age. None of them had used our technique before. Most of them were unfamiliar with auto-complete lasso.

Hypotheses: Based on the fact that our Gestalt-based technique requires much less mouse movement, we hypothesize that selection time will be shorter for this technique when selecting common salient groups. Moreover, we hypothesize that users will make fewer errors with the new technique as it conforms better to human perception.

7 Results

Selection Time: A repeated measure ANOVA revealed that technique had a strong main effect on selection time ($F_{2,20} = 40.31, p \ll 0.001$). The mean selection time for the Gestalt-based technique was 0.38 seconds, with the means for rectangle and lasso 0.84 and 1.2 seconds, respectively, see Fig. 8. A Tukey-Kramer test reveals that all three techniques are different. As illustrated in Fig. 9 and 10, orientation and shape had no significant effect on selection time, while size had a significant effect $F_{2,20} = 98.86, p \ll 0.001$. There is a strong

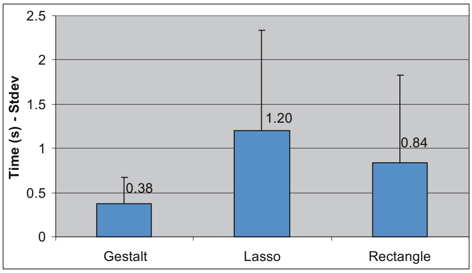


Fig. 8. Comparing selection time among techniques

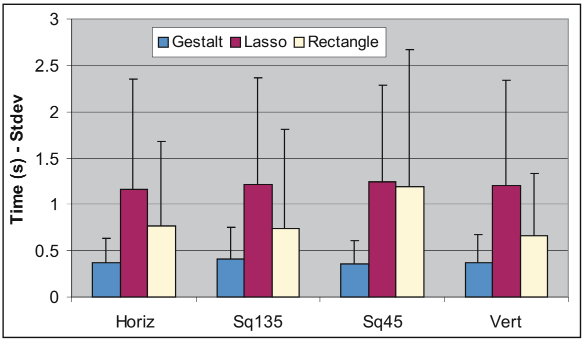


Fig. 9. Orientation does not have a significant effect on selection time

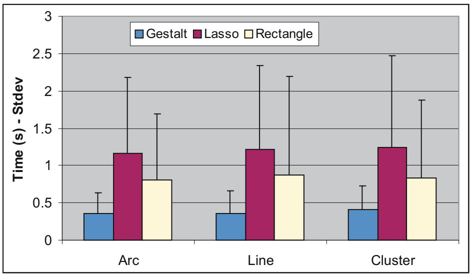


Fig. 10. Shape does not have a significant effect on selection time

interaction between technique and size of the layout, $F_{4,20} = 34.39, p \ll 0.001$. While size had no effect on Gestalt-based technique, lasso and rectangle selection time depended strongly on the size of the target group, see also Fig. 11.

Finally, we analyzed learning for each technique by plotting selection time vs. repetitions. The Gestalt-based technique showed no improvement over time, but participants got moderately faster with lasso and rectangle selection, see

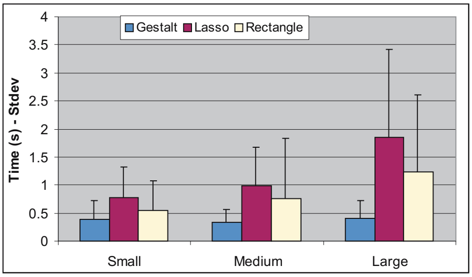


Fig. 11. Interaction between size and selection technique. Note that Gestalt is insensitive to size.

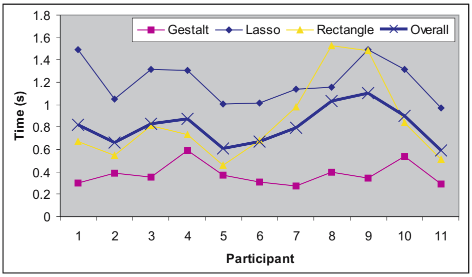


Fig. 12. Comparing overall and per technique performance of participants

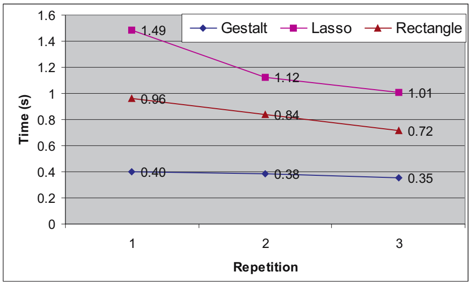


Fig. 13. Effect of repetition on selection time (learning effect)

Fig. 13. When comparing performance per technique across participants, it is notable that most variation occurred within the rectangle technique, while the Gestalt technique had the least, see Fig. 12. As there was noticeable learning observable in the experiment, we analyzed the average number of cancelations only for the final, third, repetition. For this, there is not significant difference between the cancelation rates, see also Fig. 14.

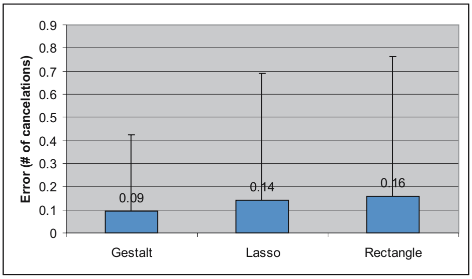


Fig. 14. Average cancelation rate for different techniques

8 Discussion

Our technique is 2.3 respectively 3.2 times faster than rectangle or lasso. This is remarkable as the there were no nearby distracters and hence the trial layouts were equally well suited for rectangle and lasso. If distracters existed, we would expect that each technique would be affected differently (e.g. rectangle selection by non-axis aligned structures, lasso by narrow tunnels, Gestalt by ambiguous cases).

Orientation has a noticeable effect on rectangle selection. In particular, the time for the sq45 layout (layouts along $y = x$) is relatively longer than all other conditions. One explanation for this is that most people draw rectangles from the top-left corner to the bottom-right. For the sq45 layouts, identifying a good top-left corner point that covers only the targets is not trivial and errorprone.

The most likely explanation for the fact that our technique is faster is that much smaller mouse movements are needed: instead of traversing the circumference of a target group or the diagonal of the corresponding bounding box, the user just (double-)clicks on one of the group members. Rectangle is faster than lasso as the diagonal is shorter than the (partial) circumference, even with auto-complete lasso. Also, when using the mouse, most people find it easier to drag out a rectangle compared to drawing a curve that traverses a tunnel. Clearly, in the presence of nearby distracters, there exist layouts and target groups where rectangle or lasso selection may perform better than our technique. For example, lasso can outperform all other techniques for partial selection of a dense, random cluster. In general, we believe that rectangle and lasso are well suited for selection of groups that have significant two-dimensional spatial extent (i.e. area arrangements) or that are axis-aligned. On the other hand, our technique is highly well suited for groups with curvilinear layouts (i.e. one-dimensional arrangements), while still being competitive for cluster selection. Hence, we believe Gestalt selection nicely *complements* rectangle and lasso selection.

9 Conclusion and Future Work

We introduced a new perceptual-based object group selection technique for spatially contiguous groups. Based on established models from perception research, we presented a new approach to automatically detect salient perceptual groups with the Gestalt principles of proximity and good continuity. Then, we introduced several new, simple, and efficient mouse-based interaction techniques to select and deselect such Gestalt groups. The results of our user study show that our technique outperforms lasso and rectangle selection when selecting groups with implicit structures.

We have not yet formally evaluated our technique in more complex scenarios, such as partial group selection and the resolution of ambiguous cases. Informal evaluations indicate that the technique works well for these cases, but we plan to do a complete user study in future work. Moreover, we will investigate extensions of our technique for highly dense configurations. Finally, we will investigate the complex interplay of spatial and similarity cues and extend our system to deal with objects with different visual features (such as shape, color, and size).

References

1. Chiu, P., Wilcox, L.: A dynamic grouping technique for ink and audio notes. In: UIST, pp. 195–202 (1998)
2. Dehmeshki, H., Stuerzlinger, W.: Using perceptual grouping for object group selection. CHI Extended Abstracts, 700–705 (2006)
3. Feldman, J.: Perceptual models of small dot clusters. DIMACS 19 (1993)
4. Feldman, J.: Curvelinearity, covariance, and regularity in perceptual groups. Vision Research 37(63) (1997)
5. Igarashi, T., Matsuoka, S., Masui, T.: Adaptive recognition of implicit structures in human-organized layouts. Visual Languages, 258–266 (1995)
6. Koffka, K.: Principles of Gestalt Psychology. Routledge and Kegan Paul (1935)
7. Kubovy, M., Holcombe, A.: On the lawfulness of grouping by proximity. Cognitive Psychology 35, 71–98 (1998)
8. Marshall, C.C., Shipman, F.M., Coombs, J.H.: Viki: spatial hypertext supporting emergent structure. Hypermedia Technology, 13–23 (1994)
9. Mizobuchi, S., Yasumura, M.: Tapping vs. circling selections on pen-based devices: evidence for different performance-shaping factors. In: SIGCHI, pp. 607–614 (2004)
10. Moran, T.P., Chiu, P., van Melle, W., Kurtenbach, G.: Implicit structure for pen-based systems within a freeform interaction paradigm. In: CHI (1995)
11. Saund, E., Fleet, D., Larner, D., Mahoney, J.: Perceptually-supported image editing of text and graphics. In: UIST, pp. 183–192 (2003)
12. Saund, E., Moran, T.P.: A perceptually-supported sketch editor. In: UIST, pp. 175–184 (1994)
13. van Oeffelen, M.P., Vos, P.G.: An algorithm for pattern description on the level of relative proximity. Pattern Recognition 16(3), 341–348 (1983)

Improving 3D Selection in VEs through Expanding Targets and Forced Disocclusion

Ferran Argelaguet and Carlos Andujar

MOVING Group, Universitat Politècnica de Catalunya, Barcelona, Spain
{fargelag,andujar}@lsi.upc.edu

Abstract. In this paper we explore the extension of 2D pointing facilitation techniques to 3D object selection. We discuss what problems must be faced when adapting such techniques to 3D interaction on VR applications, and we propose two strategies to adapt the expanding targets approach to the 3D realm, either by dynamically scaling potential targets or by using depth-sorting to guarantee that potential targets appear completely unoccluded. We also present three experiments to evaluate both strategies in 3D selection tasks with multiple targets at varying densities. Our user studies show promising results of 3D expanding targets in terms of error rates and, most importantly, user acceptance.

1 Introduction

The act of pointing to graphical elements is one of the fundamental tasks in HCI. In the 2D realm, the focus is on pointing to GUI elements such as icons, buttons and menus, whereas in the 3D realm the emphasis is put on 3D object selection. Ray-casting [1] is arguably one of the most popular virtual pointing techniques used in VEs. In the default ray-casting implementation the pointing direction is given by a virtual ray controlled by a 6-DOF sensor.

The application of Fitts' law to HCI has led to a number of successful techniques to improve pointing performance [2]. Fitts' law asserts that the time T to acquire a target of width W which lies at a distance D is governed by the relationship $T = a + b\log_2(D/W + 1)$, where a and b are empirically determined constants and the logarithmic term is called the *index of difficulty*. Fitts' law indicates three possible approaches for pointing optimization: reduce D, increase W, or a combination of both. The challenge here is how to dynamically change D and/or W *in ways that do not substantially alter the overall layout of the objects*, but nonetheless result in better performance. In the 2D interaction realm, HCI researchers have come up with a variety of pointer facilitation techniques for interacting with 2D GUI elements. Unfortunately, it turns out that devising solutions for the 3D realm is a much more involved task. For example, the extension of techniques increasing W to the 3D realm requires a proper handling of the occlusion among potential targets, which is especially difficult in the absence of a regular layout. The inclusion of constraints to guarantee a valid stereoscopic output and the lack of physical constraints involved in the use of 6-DOF sensors in free space, make the above problem even more challenging.

A. Butz et al. (Eds.): SG 2008, LNCS 5166, pp. 45–57, 2008.

In this paper we explore the extension of 2D pointing facilitation techniques to 3D object selection. We first review existing techniques and discuss what problems must be faced when adapting such techniques to 3D interaction on VR applications. In Section 3 we focus on techniques primarily increasing W, and we propose two strategies to adapt expanding targets to the 3D realm, either by dynamically scaling potential targets (Section 4) or by using depth-sorting to guarantee that potential targets appear completely unoccluded (Section 5). Three experiments to evaluate both strategies in 3D selection tasks using a 6-DOF input device are presented in Section 6. Finally, Section 7 provides concluding remarks and future work.

2 Previous Work

A review of all the strategies that have been proposed to facilitate 3D selection in VEs is out of the scope of this paper (see [3,4] for a survey). In this paper we focus on pointing facilitation techniques resulting from the direct application of Fitts' law. These techniques are orthogonal to a number of tools particularly designed for interaction in 3D space, such as physical props [5,6] and two-handed interaction.

The *optimized initial impulse* model [7] suggests that techniques attempting to improve pointing performance by decreasing D should concentrate on the initial large movement phase that covers most of the distance towards the target, whereas techniques that attempt to increase W should focus on the final corrective phase, since the effect of changes on W occurring after the large movement has been initiated will be more apparent in the final corrective phase, when the user is trying to home in on the target under closed-loop feedback control [2].

Techniques attempting to reduce D include designs where graphical elements are laid out to minimize the distance to the cursor [8]. These approaches are limited to GUI elements when potential targets are known a priori. A more general option is to reduce D only in motor space, thus preserving the original layout of the elements. This can be accomplished by ignoring the empty space between the cursor and the targets e.g. by bending the selection ray to the highest ranking object [9,10].

The second way for facilitating pointing is to increase W. This can be accomplished indirectly by increasing the size of the pointing tool, using e.g. volume cursors [11,12] or extending raycasting by replacing the selection ray by a conic volume [13]. A more direct option to control W is to increase the size of the potential targets, also known as *expanding targets*. In this situation the size of the targets or the viewing region dynamically changes to provide the user with a larger target area to interact. This technique has become popular with the MacOSX *dock* where the icons expand to a usable size when the cursor is over them. Several studies report performance improvements when using expanding targets, as the time to acquire isolated targets depends mostly on the final target size and not on the initial one [14]. The extension of these techniques to 3D interaction in VEs is discussed in Section 3.

The third family of pointing facilitation techniques attempt to both decrease D and increase W which in essence means changing the control-display (C-D) ratio. The effect of a constant C-D gain on performance has been extensively explored in the literature but results are still inconclusive [2]. The most adopted solution is to *dynamically* adjust the C-D gain [2] according to the input device speed [15] or the size of potentially selectable targets [16].

3 Adapting Expanding Targets to 3D Selection

A number of studies have demonstrated that virtual pointing often results in better selection effectiveness than competing 3D interaction metaphors such as the *virtual hand*, allowing the selection of objects beyond the area of reach and requiring less physical hand movement [3]. Nevertheless, the selection of small and distant objects has been recognized as one of the major limitations of ray-casting, especially when a 6-DOF sensor is used to control the selection ray. The acquisition of small targets require a high level of angular accuracy, which is difficult to achieve due to hand trembling and tracking errors, which are amplified with increasing distance. These involuntary movements severely disturb the final corrective movement phase, to the extent that small and partially occluded objects can be very difficult to select. As a consequence, the final corrective phase can clearly dominate the overall selection time. Furthermore, once the selection ray intersects the target, the user has to attempt to maintain the ray's orientation until the selection confirmation is triggered by, for example, pressing a button. The involuntary movement of the ray when the finger is moved to press or release the button, nicknamed the Heisenberg effect [17], introduces a further difficulty in selecting small targets. In essence, small and partially occluded objects cause (a) user dissatisfaction due to increased error rates, (b) discomfort due to the duration of corrective movements, which in the absence of physical support require an additional physical effort, and (c) unconfidence on which object will be selected after triggering the confirmation.

Since the effort to select small and partially occluded objects is governed by the final corrective phase, it seems reasonable to improve ray-casting performance by increasing W, particularly by adapting to 3D the expanding target techniques broadly used in the 2D realm. It turns out that this extension is not a trivial task. Although different distortion patterns have been studied to highlight the region of interest in 3D graphs to reduce occlusion or generate illustrations (see e.g. [18,19]), to the best of our knowledge, no controlled studies on the effect of 3D distorsion viewing on selection performance have been conducted. We have identified the following problems in adaptating expanding targets to 3D:

- The extension to 3D scenes requires a proper handling of the occlusion among potential targets. If an object is expanded, it can potentially occlude neighboring targets which were visible before the expansion. Unlike the 2D counterpart, occlusion among 3D objects is view-dependent and is a *global*

problem, meaning that two objects arbitrarily far away in 3D can occlude each other when the user's viewpoint is aligned with them.
- Occlusion handling is especially difficult in the absence of a regular layout. Unlike GUI elements, objects in a 3D scene exhibit no regular arrangement.
- To ensure the compatibility with VR display systems, correct parallax values on the projected stereo images are required. This means that techniques based on 2D image distortion [20] cannot be used in VR systems as they completely disrupt parallax values.

Besides these problems, there are other factors that come into play when selecting 3D objects. On the one hand, 3D targets appear at a variety of scale levels. In a typical 2D GUI there is a minimum size for all interface elements. However, in a 3D scene viewed by a moving observer, there is no lower limit to W. As a consequence, expanding targets can be desirable not only to facilitate pointing to small targets, but also to help the user to visually recognize the object pointed to as the intended target. On the other hand, the frequency of pointing acts follows a different pattern in a 2D GUI application compared to a VR application. In the former, pointing is the most fundamental task and selection times have a substantial impact on user productivity. In VR applications, pointing acts are mixed with navigation and manipulation tasks, suggesting that user satisfaction should be a primary goal [3].

We consider two orthogonal methods to increase W in a 3D scenario: we can scale the potential targets (thus increasing their size and hence their screen projection) and we can maximize the number of visible pixels of the potential target by using depth-sorting to guarantee that potential targets appear completely unoccluded. In the next two sections we detail two possible implementations of both approaches.

4 Increasing W through Dynamic Scaling

The main idea is to expand objects near the selection ray to facilitate their selection. The object intersected by the selection ray will be referred to as the *focus*. We first discuss how to scale the focus and then how to propagate the transformation to other objects. We assume that a usable size for the focus (in terms of the solid angle subtended from the viewpoint) has been decided considering the characteristics of the input device and the user preferences. This size can be dynamically converted into a number of pixels Π considering the screen resolution and the current viewpoint. The scale factor for the focus can be computed as $s_f = \max(1, \sqrt{\Pi/P_i})$ where P_i is the number of pixels in the screen projection of the (unexpanded) focus. P_i can be easily computed using OpenGL's occlusion queries, and it only has to be recomputed for significant viewpoint changes. The computed s value is used to scale the focus in 3D space with respect to its center, see Figure 1(a).

When an object is expanded it can potentially occlude neighboring targets, thus requiring some mechanism to guarantee that originally-visible objects can still be selected after the expansion. This condition must be enforced at least

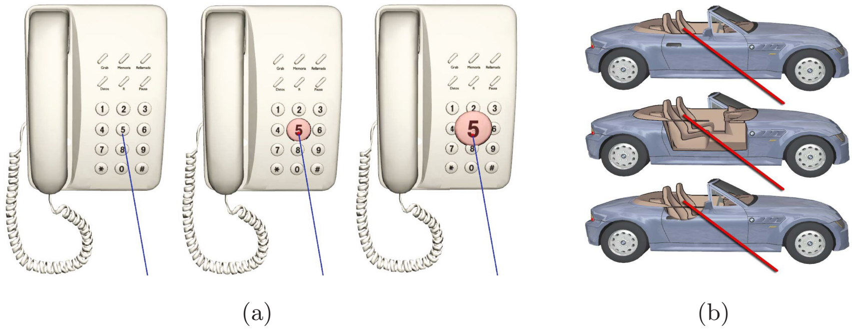

(a) (b)

Fig. 1. Adapting expanding targets to 3D: (a) dynamic scaling; (b) forced disocclusion

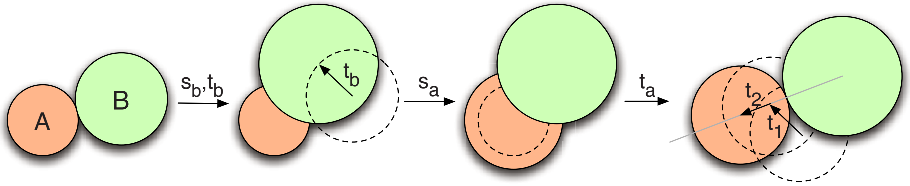


Fig. 2. Propagation of transformations. The transformation (s_b, t_b) applied to object B is used to compute the transformation (s_a, t_a) to be applied to object A.

in the vicinity of the ray. Although all object transformations will be applied in 3D space, potential overlaps among objects must be analyzed in 2D considering their screen projection. Potential overlaps from a given viewpoint can be encoded by a graph G where nodes represent objects and there is a link joining A and B if the screen projection of A can overlap with that of B after a transformation. Figure 3(d) shows an example. Objects outside the viewing frustum and completely hidden objects need not to be represented in the graph. Graph G is obviously view-dependent and can be recomputed every time the viewpoint moves far apart. Since the viewpoint can be considered to be roughly stationary during selection, this means that G must be recomputed only when the viewpoint stabilizes, indicating the potential start of a pointing act. The graph could have $O(n^2)$ links, but we have observed a linear behavior in all the scenes we tested with reasonable Π values. A conservative way for computing G is to consider the screen projection of the object's bounding spheres.

The transformation for the rest of the objects can be encoded as $(s, \boldsymbol{t})$ where s is the scaling factor wrt the object's center, and $\boldsymbol{t}$ is a translation vector. These values can be computed by a local breadth-first traversal of G starting from the focus object. The traversal can be restricted to a maximum depth to limit its effect to the objects in the vicinity of the ray. For each visited object A, we search the graph for the neighboring object B maximizing the overlap with A. The search is limited to already visited objects for which the transformation has been already computed. If the overlap is below a certain threshold, the transformation $(s_a, \boldsymbol{t_a})$ for object A is left unmodified. Above this threshold, the

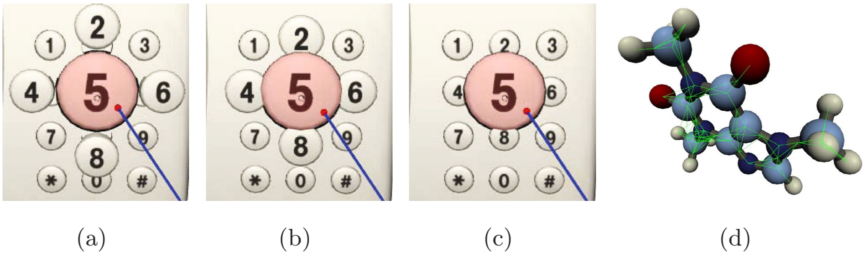


Fig. 3. Different overlaps can be considered when propagating transformations: (a) no overlap is allowed; (b) ≈ 25% overlap; (c) ≈ 50% overlap. Image (d) shows the molecule model and its view-dependent graph.

transformation for A can be computed as depicted in Figure 2. Let r_a and r_b be the radius of the bounding circle of the screen projection of A and B, as they appear in the original, untransformed scene. After scaling B by s_b, the radius of B is incremented by $\Delta r_b = r_b(s_b - 1)$. It seems reasonable to compute the scale factor for A with the constraint $\Delta r_a = \Delta r_b$. This can be accomplished by letting $s_a = (\Delta r_b / r_a) + 1$. In the experiments we used this approach for computing s_a in combination with a linear attenuation based on the distance to the focus object. The translation vector $\boldsymbol{t_a}$ can be computed as $t_1 + t_2$, where $t_1 = t_b$ simply propagates to object A the translation applied to object B, and t_2 is computed so that the screen projection of object A is moved apart from that of B while preserving their relative overlap in the original, untransformed scene. The direction of t_2 is defined by the line joining the centers of their screen projections. The computation of t_2 can be based on different overlap values, as shown in Figure 3. In the experiments we allowed for a 50% overlap.

The avoid abrupt changes, these transformations can be applied gradually as shown in Figure 1(a). Every time a focus change occurs, the transformation for all objects is computed and it will be smoothly interpolated across several frames (during 35ms in the experiments). If another focus change occurs while still playing the animation, the new transformation values should be computed from the interpolated values at the time of the event.

5 Increasing W through Forced Disocclusion

An orthogonal approach to increase W is to maximize the number of *visible* pixels of the focus object by forcing it to appear completely unoccluded. This can be easily accomplished by a proper use of OpenGL's depth range, see Figure 1(b)-middle. To avoid excessive occlusion of neighboring objects, we can force disocclusion only in a circular region of Π pixels around the intersection point, as in Figure 1(b)-bottom. This can be easily implemented e.g. using the stencil buffer and only requires a second rendering pass for the focus object.

Forced disocclusion increases W also in control space if the ray-scene intersection is adjusted to match object visibility. So far, when the selection ray

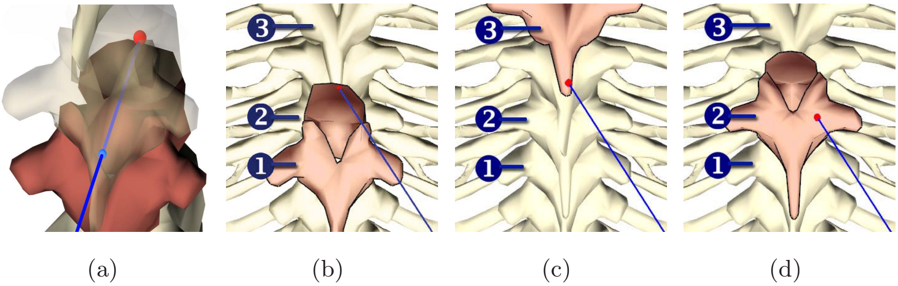


Fig. 4. The use of forced disocclusion worsens the problem caused by visibility mismatch. Object 1 is partially occluded by other objects, as shown in *(c)*. Suppose now the user selects Object 1. The resulting image is shown in *(b)*, where Object 1 has been drawn completely unoccluded. The ray intersects Object 2 first, but this intersection is ignored to match the visual feedback, as discussed in the text. Now suppose the user moves the ray slightly upwards. Visually, the expected result is that the next object to be selected should be object 3, as shown in *(c)*. However, object 2 will be the next selected object *(d)*, because as soon as the ray leaves object 1, the ray still intersects object 2 and this intersection is no longer ignored *(a)*.

intersected several objects, we considered only the first intersection. Now we must guarantee that the focus object will remain selected until the ray leaves the object, *ignoring any potential closer intersections*. This situation is depicted in Figure 1(b)-middle, where the intersection with the (hidden) door must be ignored until the ray leaves the selected object. This selection behavior is straightforward using OpenGL's selection buffer with different depth ranges.

We conducted a pilot study to get a first evaluation of the technique as described above, using a 6-DOF sensor to control the ray. Interestingly, most users reported an erratic behavior of the ray when interacting with densely-occluded scenes. We discovered that the reason for this behavior was due to the fact that the user's eye and the user's hand (used as the origin of the selection ray) might "see" a different subset of objects. This visibility mismatch between the user's hand and the user's eye occurs whenever the ray is controlled by the hand. However, its effects are much more apparent when forced disocclusion is enabled. This situation is depicted in Figure 4 and results in a completely counterintuitive behavior.

This problem can be ameliorated by further adapting the ray-scene intersection. First we compute the intersection of the current ray with the scene. If the ray has left the object that was selected in the previous frame, we proceed as follows. Let P_{old} be the intersection point with the object selected in the previous frame. Since we are forcing disocclusion, P_{old} is not guaranteed to be the closest intersection, as shown in Figure 5(a). We compute a second ray defined by joining the *user's eye* position with P_{old}. The closest intersection P_{new} of this second ray with the scene indicates the next selected object, and we introduce a slight correction in the orientation of the ray so that it intersects P_{new}. This correction means that we create a subtle mismatch between the orientation of

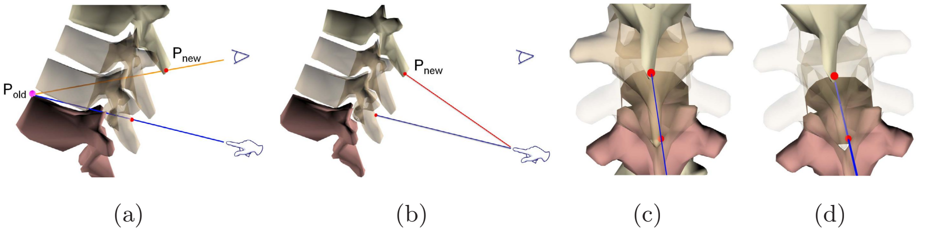


Fig. 5. Modified intersection test used with forced disocclusion. Starting from the situation in (a), if the ray is moved upwards, the next selected object will be that indicated by P_{new}, which matches with the behavior expected from the user's viewpoint. The orientation of the ray is modified to intersect P_{new} (b). The resulting orientation offset between the ray and the device is hard to notice from the user's viewpoint (c-d).

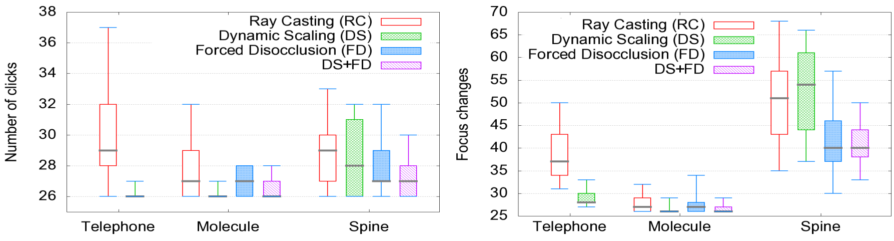
the input device and the ray orientation, as shown in Figure 5(b). The resulting offset is similar to the offset accumulated by techniques adjusting C-D gain according to speed movement [15]. The orientation disparity occurs in the plane defined by the ray and the user's eye, and thus it is hardly visible from the user viewpoint, see Figures 5(c) and 5(d). This offset will be accumulated until a fast movement of the user's hand is detected (30 deg/s was used in the experiment).

6 Experiment

Given that the effort to select small and partially occluded objects in the default ray-casting implementation (RC) is governed by the final corrective movements, in the best case scenario one could expect *dynamic scaling* (DS) and *forced disocclusion* (FD) to have a positive impact in selection performance. In practice, however, this may not be the case. On the one hand, the movement of neighboring targets to avoid occlusion with expanded targets could be potentially distracting to the users and negate the benefits of DS. On the other hand, forced disocclusion of the focus object might occlude neighboring objects and result in poor performance. The main goal of our experiments is therefore to evaluate potential advantages of DS and FD in selection time, error rates, user discomfort, and user confidence, in scenes with multiple targets at varying densities.

Apparatus. All the experiments were conducted on a four-sided CAVE with active-stereo projectors at 1280x1280 resolution. The input device was a 6-DOF Ascension Wanda and a tracking system with 2 receivers providing 60 updates/s with 4 ms latency. At the user position (90cm from the EM emitter), position and orientation RMS errors were below 0.5mm and 0.06 degrees, resp. The experiment was driven by a cluster of 2.66GHz QuadCore PCs with GF8800 GTX cards.

Participants. Sixteen volunteers (4 female, 12 male), aged from 24 to 41, participated in the experiment. Most participants (9) had no experience with VE applications; 5 had some experience and 2 were experienced users.



(a)

(b)

Fig. 6. Box plots for number of clicks (a) and focus changes (b). For the telephone model, only DS was considered as a factor.

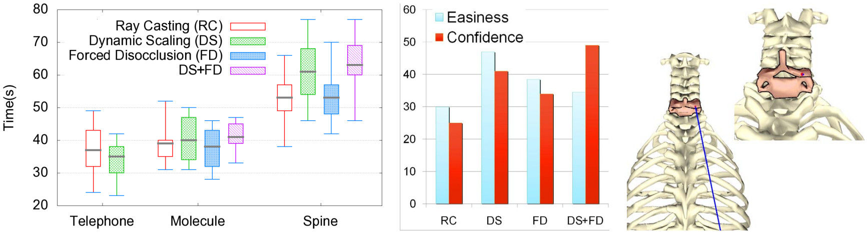
Procedure. The task was to select a sequence of objects, where the next object to be selected was clearly highlighted. Visual feedback was provided also for the object having the focus which was highlighted in red. We also provided acoustic feedback by triggering a different sound every time users hit or missed a target. Users were requested to complete the selection task as accurately as possible, trying to press the wanda button once per target. We used three different models with varying density and occlusion levels (see the accompanying videos [21]). The models were placed at 3m from the user position.

Design. A repeated-measures within-subjects design was used. The independent variables were DS (enabled, disabled) and FD (enabled, disabled), resulting in four combinations. Each participant performed the experiment in one session lasting approximately 25 min. The session was divided into four blocks, one for each technique. Before each block users were provided with a short training session which required them to complete practice trials. Each participant did the four blocks in a random order.

The dependent measures were total selection time, homing time, error rate and focus changes. We measured homing time from the first intersection of the ray with the target till the button press. The error rate was measured by counting the number of erroneous clicks. The focus changes was the number of times the target object changed its selection status prior to confirming the selection.

Results. The goal of the first experiment was to evaluate the effects of expanding targets in a 3D scene with low target density. We chose the telephone model shown in Figure 1(a). Since all targets appeared already unoccluded, only DS was considered as a factor. Results of the experiment are shown in Figures 6 and 7(a). The one-way ANOVA showed a significant effect ($\alpha = 0.05$) for DS in error rate ($p < 0.01$), homing time ($p < 0.05$), and focus changes ($p < 0.01$). In all these measures users performed significantly better with DS enabled. This result was expected as in the telephone model potential targets are relatively far apart and benefits of enabling DS on accuracy are more apparent.

The molecule model, shown in Figure 3(d), was chosen as a representative of a scenario with multiple targets at close proximity. We considered both DS and FD as factors. The two-way ANOVA showed a significant effect for DS in



(a) (b) (c)

Fig. 7. Box plot of the selection time (a); Friedman rank test on easiness and confidence (b); Difference between the scene as seen from user viewpoint and from the user's hand. Note that the hole is only visible from the hand position.

error rate ($p < 0.01$) and focus changes ($p < 0.01$), again users performing significantly better with DS enabled. No significant differences were found in terms of selection time and homing time for DS, FD and DSxFD ($p > 0.3$). This suggests that DS benefits on accuracy also apply to scenes with potential targets located in close proximity. The lack of a significant effect for FD was also expected considering the relatively low level of occlusion between targets.

The last model, shown in Figure 7(c) was selected to study the effects of DS and FD on a worst-case scenario with multiple potential targets with a high degree of overlap. In this case the two-way ANOVA showed a significant negative effect for DS in selection time ($p < 0.01$), and homing time ($p < 0.01$), whereas we found no significant effect for DS in error rate ($p > 0.3$). Conversely, a positive effect was found for FD in error rate ($p < 0.04$) and focus changes ($p < 0.01$), see Figure 6. No significant effect was found for the interaction DSxFD in any of the dependent measures. The analysis on a per-object basis revealed that two vertebrae were particularly difficult to select with DS enabled. We discovered that the common point on these two objects was that, when expanded, their through hole was *visible* from the user's hand but not from the user's eye. This situation is depicted in Figure 7(c). Our best explanation for the poor behavior of DS with these two objects is that the following sequence of events was repeated multiple times until the selection could be confirmed by the user: (1) the target was intersected by the ray, thus triggering its expansion; (2) the expansion caused the through object to be exposed, potentially causing the selection ray to go through the hole and thus missing the object; (3) once the object lost the focus, the object smoothly returned to its original size. We observed that this loop was repeated several times for these objects. The fact the hole was not visible for the eye was another major factor that came into play to make selection more difficult. This behavior seems to be confirmed by the abnormally high number of focus changes.

Survey. After each block participants were requested to rate the *easiness* of the selection, the *confidence* on hitting the right target immediately before pressing the button, and the level of *physical effort*, using a 7-point Likert scale. We found significant differences for the easiness and confidence, but not for the effort level. Figure 7(b) shows the ranks obtained by the Friedman test. Users found selection significantly easier with DS enabled ($p < 0.05$), followed by FD, DS+FD and RC. Concerning confidence on hitting the target, the Friedman test also indicated a significant difference ($p < 0.01$), ranking DS+FD in the first place, followed by DS, FD and finally RC. This seems to confirm that, since the selection of difficult targets is governed by the final corrective phase, dynamic scaling is perceived by the user as a positive facilitation technique.

Technique overhead. We also measured the performance overhead of DS and FD. All overheads were found to be negligible with the only exception of the graph computation, which nevertheless was computed in less than 5ms for moderately-sized scenes. Since this graph needs to be recomputed only when the viewpoint changes significantly, the impact on the frame rate can be neglected.

7 Conclusions and Future Work

We have presented dynamic scaling and forced disocclusion, two pointing facilitation techniques resulting from the adaptation of 2D expanding targets to 3D selection on VEs. Should time performance be the primary goal, the two presented embodiments do not provide a significant advantage. As with most pointing facilitation techniques, time improvements are more apparent in situations where targets are isolated. When targets are more closely packed together, the benefit of these techniques tend to degrade, and can even be detrimental to selection time. This does not mean, though, that 3D expanding targets do not offer advantages. In the context of real usage in a VR application, the subjective impressions of an interaction technique can play a much larger role than speed in controlled experiments. The inability to control accurately the selection ray may prove to be overly annoying to the user and thus be a source of dissatisfaction. We have already pointed out the benefits of 3D expanding targets in terms of error rates. By focusing in the final movement phase, the proposed techniques help the user to keep the selection ray over the intended object and confirm the selection with more certainty. Being less sensible to involuntary movements, expanding targets can be particularly suitable when the selection ray is controlled with a 6-DOF sensor in free space. The enlargement in visual space also facilitates the visual recognition of the object, an important aspect which is lacking in techniques manipulating the C-D ratio. Most importantly, our user studies showed promising results of 3D expanding targets in terms of user confidence and user acceptance.

There are several directions that can be pursued to extend the current work. For densely-occluded scenes, it may be interesting to experiment with alternative ways for moving objects apart. Concave objects and objects with through holes pose a problem with dynamic scaling, as the ray might no longer intersect the object after the expansion. It would be useful to modify the intersection test to avoid such a distracting effect. It may be interesting to explicitly evaluate the effect of partial disocclusion on depth perception, evaluating the potential conflict between occlusion and binocular depth cues.

Acknowledgements. This work has been partially funded by the Spanish Ministry of Science and Technology under grant TIN2007-67982-C02-01.

References

1. Bowman, D., Hodges, L.: An evaluation of techniques for grabbing and manipulating remote objects in immersive virtual environments. In: I3D 1997: Proceedings of the 1997 ACM Symposium on Interactive 3D graphics, pp. 35–38 (1997)
2. Balakrishnan, R.: "beating" fitts' law: Virtual enhancements for pointing facilitation. International Journal of Human-Computer Studies 61(6), 857–874 (2004)
3. Bowman, D.A., Kruijff, E., LaViola, J.J., Poupyrev, I.: 3D User Interfaces: Theory and Practice. Addison Wesley, Reading (2004)
4. Steed, A.: Towards a general model for selection in virtual environments. In: IEEE Symposium on 3D User Interfaces (2006)
5. Lindeman, R., Sibert, J., Hahn, J.: Towards usable VR: an empirical study of user interfaces for immersive virtual environments. In: Proceedings of the SIGCHI conference on Human factors in computing systems, pp. 64–71 (1999)
6. Schmalstieg, D., Encarnacao, M., Szalavri, Z.: Using transparent props for interaction with the virtual table. In: Proceedings of the 1999 symposium on Interactive 3D graphics, pp. 147–153 (1999)
7. Meyer, D., Abrams, R., Kornblum, S., Wright, C., Smith, J.: Optimality in human motor performance: ideal control of rapid aimed movements. Psychological Review 95(3), 340–370 (1988)
8. Dachselt, R., Hübner, A.: Three-dimensional menus: A survey and taxonomy. Computers and Graphics 31(1) (2007)
9. Steinicke, F., Ropinski, T., Hinrichs, K.: Object selection in virtual environments with an improved virtual pointer metaphor. In: International Conference on Computer Vision and Graphics, ICCVG 2004 (2004)
10. de Haan, G., Koutek, M., Post, F.: IntenSelect: Using Dynamic Object Rating for Assisting 3D Object Selection. In: EG workshop on VEs (2005)
11. Zhai, S., Buxton, W., Milgram, P.: The silk cursor: investigating transparency for 3d target acquisition. In: CHI 1994: Proceedings of the SIGCHI conference on Human factors in computing systems, pp. 459–464 (1994)
12. Vanacken, L., Grossman, T., Coninx, K.: Exploring the effects of environment density and target visibility on object selection in 3d virtual environments. In: IEEE Symposium on 3D User Interfaces, 2007. 3DUI 2007 (2007)
13. Forsberg, A., Herndon, K., Zeleznik, R.: Aperture based selection for immersive virtual environments. In: User interface software and technology, pp. 95–96 (1996)
14. McGuffin, M., Balakrishnan, R.: Acquisition of expanding targets. In: CHI 2002: Proc. SIGCHI conference on human factors in computing systems, pp. 57–64 (2002)
15. Frees, S., Kessler, D.: Precise and rapid interaction through scaled manipulation in immersive virtual environments. In: Proc. IEEE Virtual Reality, pp. 99–106 (2005)
16. Andujar, C., Argelaguet, F.: Anisomorphic ray-casting manipulation for interacting with 2d guis. Computer Graphics 31(1), 15–25 (2007)
17. Bowman, D., Wingrave, C., Campbell, J., Ly, V.: Using pinch gloves for both natural and abstract interaction techniques in virtual environments. In: HCI International, pp. 629–633 (2001)

18. Carpendale, M., Cowperthwaite, D., Fracchia, F.: Extending distortion viewing from 2d to 3d. IEEE Comput. Graph. Appl. 17(4) (1997)
19. Raab, A., Rüger, M.: 3d-zoom: Interactive visualisation of structures and relations in complex graphics. In: 3D Image Analysis and Synthesis, pp. 125–132 (1996)
20. Germer, T., Götzelmann, T., Spindler, M., Strothotte, T.: Springlens - distributed nonlinear magnifications. Eurographics Short Papers (2006)
21. `http://www.lsi.upc.edu/~virtual/ET`

Finger Walking in Place (FWIP): A Traveling Technique in Virtual Environments

Ji-Sun Kim[1], Denis Gračanin[1], Krešimir Matković[2], and Francis Quek[1]

[1] Virginia Tech, Blacksburg, VA 24060, USA
hideaway@vt.edu, gracanin@vt.edu, quek@vt.edu
[2] VRVis Research Center, Vienna, Austria
matkovic@vrvis.at

Abstract. In this paper we present a Finger Walking in Place (FWIP) interaction technique that allows a user to travel in a virtual world as her/his bare fingers slide on a multi-touch sensitive surface. Traveling is basically realized by translating and rotating the user's viewpoint in the virtual world. The user can translate and rotate a viewpoint by moving her/his fingers in place. Currently, our FWIP technique can be used to navigate in a plane but it can be extended to navigate in the third axis, so that the user can move to any direction in a 3D virtual world. Since our FWIP technique only uses bare fingers and a multi-touch device, finger motions are not precisely detected, especially compared with the use of data gloves or similar sensing devices. However, our experiments show that FWIP can be used as a novel traveling technique even without accurate motion detection. Our experiment tasks include finding and reaching the target(s) with FWIP, and the participants successfully completed the tasks. The experiments illustrate our efforts to make the FWIP technique robust as a scaled-down walking-in-place locomotion technique, so that it can be used as a reliable traveling technique.

Keywords: Virtual environments, Finger-walking, Navigation, Traveling techniques, Multi-touch device.

1 Introduction

One of the main interaction tasks in virtual environments (VEs) is navigation. The interaction techniques are based on input devices [3] specific to VE systems or simulators [4,10]. Slater et al. [9] state that more natural locomotion enhances the sense of presence in VEs. We can assume that users may have the better spatial awareness because natural locomotion can give more sensory cues such as proprioception, vestibular apparatus, and kinaesthetic sense as well as vision that would help them get the spatial knowledge [8]. Several interaction techniques have been developed for different types of the navigation tasks, to effectively support natural locomotion in VEs [3].

One classification of traveling techniques is based on the overall interaction metaphor (e.g. physical locomotion, steering, route-planning, target-based,

A. Butz et al. (Eds.): SG 2008, LNCS 5166, pp. 58–69, 2008.

manual manipulation, and scaling) [3]. If we see the interaction techniques developed with a metaphor, especially “physical locomotion”, we notice that the metaphor is straightforwardly transferred from the real world to the virtual world. For example, walking locomotion techniques, such as “real walking”, “walking in place”, and “simulated walking” [4,9,10], are only realized with physical walking motion. On the other hand, most of the interaction techniques with the steering metaphor (e.g. gaze, pointing, or steering props) are transformed to the flying locomotion technique. While the natural walking locomotion has been applied to get better spatial knowledge [5,8], the users can get less benefits (due to physical body fatigue), compared with the use of the steering metaphor. On the other side, the steering metaphor based techniques provide easy ways to navigate a virtual world, but the effects on the spatial knowledge acquirement are smaller compared with the natural locomotion techniques [5,8].

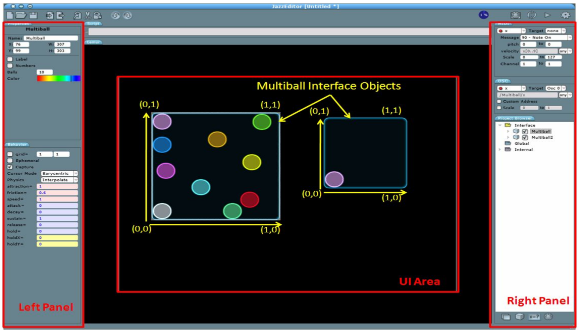
We provide a traveling technique, named Finger Walking in Place (FWIP), to take advantages of walking motion and steering control by fingers. It is a walking metaphor based interaction technique *transformed* from one of physical walking motion, i.e. “walking in place”, to the finger walking motion. It is realized with bare fingers on a physical surface. Users can move forward and backward, and rotate in a virtual world. The users actually *feel* traveling in the virtual world by walking. It allows users to control virtual walking speed, i.e. by controlling the distance and the frequency of the finger movement, as the same way in their physical walking by controlling the distance and the frequency of the leg movement. The sense of presence is reduced with FWIP as “walking in place” does not provide the real vestibular cues [3]. However, the physical motion with FWIP is scaled-down, and consequently physical body fatigue should be diminished, compared with the physical locomotion technique.

If fingers represent human legs, walking motion using fingers should be able to translate the viewpoint in the virtual world. In a technical way, the viewpoint translation can be implemented with a single-touch device using the sliding motion of fingers as human legs slide on a treadmill because two fingers need not to touch the device surface at the same time. However, it is almost impossible to fully rotate one’s hand on the surface to rotate the virtual world in place. There are two ways to control the viewpoint; physical and virtual techniques [3]. The former means that a user physically moves to translate or rotate the viewpoint, while the latter means that a user’s body remains in place to control the virtual viewpoint. Since our technique is transformed from a physical locomotion technique and is realized with fingers only, we assume that the user’s body remains in place during the navigation. Hence, although the full rotation with the user’s body is possible in the physical technique, the virtual technique realized with fingers has a limitation in terms of the rotation angle. In order to allow the user to freely control both translation and rotation of the viewpoint with fingers, we need to consider different motions for the viewpoint rotation from the sliding walking motion. Consequently, we decided to use a multi-touch sensitive surface.

1.1 Multi-touch Devices

Currently, the general multi-touch techniques are becoming more available for various device sizes from a palm size (e.g. handheld devices) to a table (e.g. Microsoft Surface) or wall size (e.g. SensitiveWall) displays. The applied technologies vary from simulating with computer vision to using resistive or capacitive touch screens. For example, one of our previous work [7] used an Augmented Reality technology using computer vision and a regular tabletop surface for navigation task in VEs. To the best of our knowledge, the first commercial multi-touch display device was Lemur [6] by JazzMutant in 2005, followed by Apple's iPhone [1] in 2007. Other devices followed, including Dexter by JazzMutant and iPod Touch [2] by Apple. As touch display technology advances from single-touch to multi-touch sensing, the number of interaction techniques is growing. For example, one of the most popular interaction techniques is a two-finger interaction technique applied to mobile handheld devices (e.g. iPhone and iPod Touch) for zooming and resizing tasks.

We chose Lemur [6] to implement our technique because it already comes with JazzEditor which provides a user interface (UI) development environment, while the Software Development Kit (SDK) for iPhone has been published on March 2008 after we implemented our technique with Lemur. Lemur is originally developed for MIDI applications. It provides a multi-touchable 12" display surface (800 x 600 pixels resolution) and completely customizable screens that are editable with JazzEditor. A designer can freely position UI objects, e.g. Container, Monitor, Multiball, Fader, Switches, and so forth, on the screen and easily change their properties such as size, color and inertia effect. The coordinate system used for the UI in JazzEditor is localized to each object placed on the screen. For example, Multiball, which we used to implement our technique, is an x-y rectangle area controller that can have up to ten controllable balls. Regardless of the size of the rectangle, the left-bottom corner is the origin (i.e. 0, 0) and the scale is up to 1, as shown in Figure 1(UI area). Lemur can recognize up to ten different spots at once. When a user places her/his fingers on the MultiBall object area, the ball nearest the spot touched becomes the one that she/he controls. The Lemur uses Open Sound Control (OSC) messages to communicate with a host



Multiball
Name: Multiball
X: 76 W: 307
Y: 99 H: 303
□ Label
□ Numbers
Balls 10
Color

□ grid= 1 1
□ Ephemeral
☑ Capture
Cursor Mode Barycentric
Physics Interpolate
attraction= 1
friction= 0.6
speed= 1
attack= 0
decay= 0
sustain= 1
release= 0
hold= 0
holdX= 0
holdY= 0

● x Target none
Message 90 - Note On
pitch 0 to 0
velocity x[0..9] any
Scale 0 to 127
Channel 1 to 1

● x Target Osc 0
/Multiball/x any
□ Custom Address
□ Scale 0 to 1

Interface
☑ Multiball
☑ Multiball2
Global
Internal

Fig. 1. JazzEditor: a property panel (left), UI area (center), and a message panel (right)

computer over Ethernet. The properties of each UI object and the messages to the host computer can be customized and edited (Figure 1).

2 Design

VT-CAVE [11] installation driven by Linux machines is used in this study, including a Fakespace 4-wall CAVE display, which is 10x10 feet long, and an Intersense IS-900 VET tracking system for a head tracker, which is attached on the shutter glasses. The Lemur is placed on the table and its position is fixed in the middle of the CAVE immersive space (Figure 2). As long as the Lemur stays in place, its position acts as a persistent spatial reference.

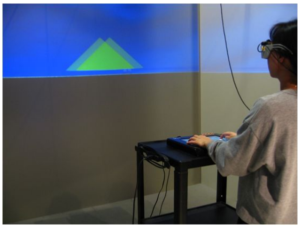


Fig. 2. Experiment setup for FWIP

2.1 Interaction Technique

We provide three types of basic functions; walking in place to walk forward, backward and turn, rotation in place, and control of the walking speed.

Walking in place: In order to move forward (Figure 3(a)), one finger first touches the surface and slides down while touching the surface, as if human legs move on a treadmill. The finger then leaves the surface and immediately the next finger (or the same finger) touches the surface and slides down. Thus, a user can move forward by repeating this process. Figure 3(b) shows that the user can move forward, changing the viewpoint by turning the hand. When moving backward, one finger first touches the surface, slides up, and leaves the surface (Figure 3(c)). The user can freely walk forward and backward by changing the sliding motion.

Rotation in place: Since it is difficult to turn the hand in place for more than 60–90 degrees, we designed three rotation-in-place techniques that can be performed on a different area separately from the walking area on the device surface. The rotation-in-place techniques are only used to rotate the viewpoint while the user's position does not change in the virtual world. In order to simultaneously translate and rotate the viewpoint, the user needs to use two hands, one for walking and another for rotation. The first one, Figure 4(a), mimics the walking motion for turning. The angle for the viewpoint rotation is changed by

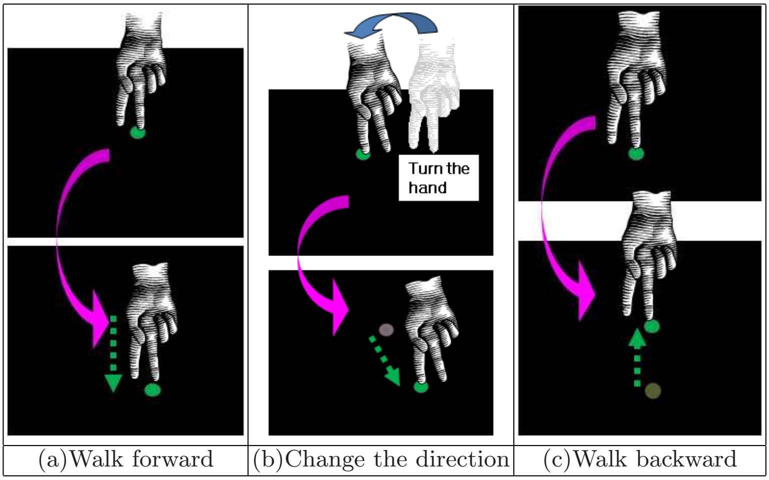


Fig. 3. Walking in place

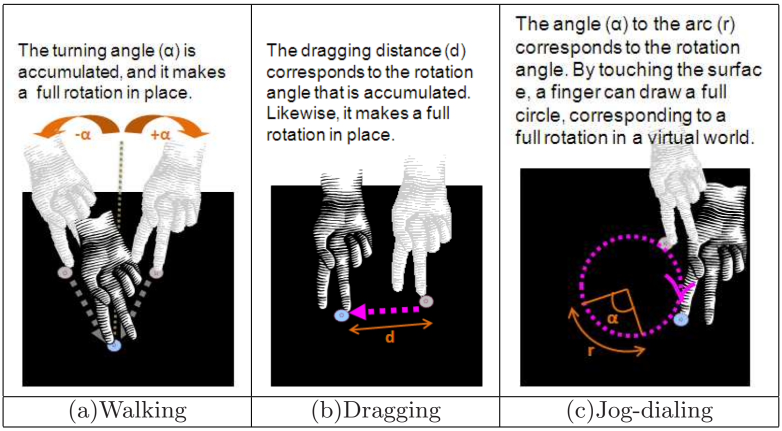

The turning angle (α) is accumulated, and it makes a full rotation in place.
-α
+α
The dragging distance (d) corresponds to the rotation angle that is accumulated. Likewise, it makes a full rotation in place.
d
The angle (α) to the arc (r) corresponds to the rotation angle. By touching the surfac e, a finger can draw a full circle, corresponding to a full rotation in a virtual world.
α
r
(a)Walking
(b)Dragging
(c)Jog-dialing

Fig. 4. Rotation-in-place technique

the angle which is formed by turning the hand. The second design, Figure 4(b), is based on the traditional mouse technique to change the user's perspective when the user navigates a 3D virtual world using a keyboard and a mouse. The longer dragging with a mouse makes the larger change for the user's perspective. Likewise, the dragging distance with a finger corresponds to the rotation angle in our technique. The third one, Figure 4(c), is designed from the finger technique of the traditional remote control device to slowly play a video. This technique can be more intuitively applied for the viewpoint rotation in that the viewpoint rotation corresponds to the circle angle. However, for this paper, only first two techniques are evaluated due to time constraint.

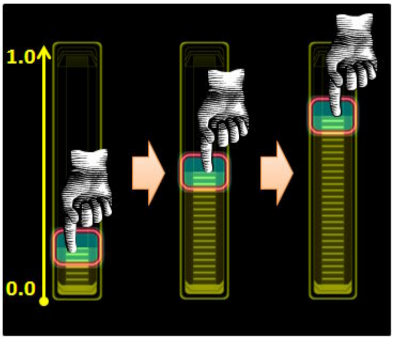


Fig. 5. Walking speed control example: speed-up

Control of the walking speed: Since the basic algorithm for walking is designed with a vector defined by two touch spots, the walking speed is basically controlled by the vector length (i.e. finger sliding length) in addition to the frequency of the finger movement. As shown in Figure 5, Fader object can be used for controlling the walking speed. The fader object is resizable, but its value is always ranged from (0.0) to (1.0), according the local coordinate system. When its control pointer is at (0.0), the walking speed is only controlled by the finger movement. With the current algorithm, as its control pointer is going up to (1.0), the walking speed is increasing up to twice the original speed.

3 Evaluation

3.1 Virtual World

The virtual world is designed to have no visual aid for finding a path to reach a target point. The space boundaries are fixed with the 40x40 meters and displayed using small cylinders. A green cone object represents the starting point. A red cone is placed behind of the starting point, so that we can examine which one a

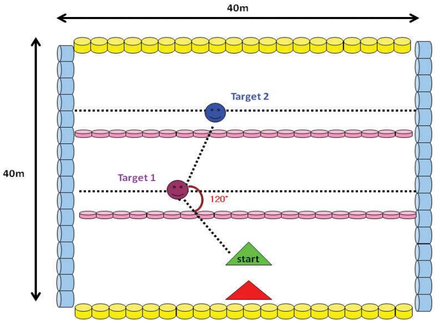


Fig. 6. Virtual World

user would prefer, walking backward or rotation in place, to reach it. In order to trigger the user to find the target points randomly placed with the 120 degree angle (Figure 6), two lines are displayed at 10m and 20m from the starting point.

3.2 Pilot Study

We performed an initial study with five participants to evaluate the FWIP technique. The experimental tasks included finding one or two targets in near or far distances from the starting point. In this study, the rotation-in-place technique was not provided. We also asked the participants to use a joystick-based traveling technique for the same tasks. They were asked to fill out the pre-experiment questionnaire which includes demographic questions, such as age, gender, frequency of playing computer or video games, and VE experience level. Since they were all novice VE users, they had to be trained with both user interfaces before starting the experiment. A post-experiment questionnaire obtained subjective preference of two interfaces as well as free-form comments. All of them were interested in using both user interfaces, but more preferred the joystick technique to complete the experiment tasks because flying locomotion is continuous and faster than walking locomotion. In addition, there was the latency issue between a user's input and the visual feedback. We also observed that most of participants occasionally moved out of the touch area while they were moving their fingers without looking at their finger movement to navigate the virtual world.

Enhancement

- **Latency:** To address the latency issue uncovered in the pilot study we made a change of the finger walking algorithm specific to the Lemur device to improve usability. While the original algorithm was based on the simple vector of the first and last touch spots per sliding, the improved algorithm is based on two consecutive touch spots. We also adjusted the angle of the vector formed by two spots. For this, we used the walking pattern acquired from the pilot user study. Figure 7 shows the partial data which the Lemur sends to the host computer. We analyzed x and y values and found that the y-range depends on the walking area size. We also found that the x-range is slightly changed even though users think they are walking straight to the target. These findings were applied to change the algorithm for the second experiment.
- **Rotation techniques:** We used our two rotation-in-place techniques after the first user study was conducted. While the first user interface is designed with one Multiball object (Figure 8(a)), the second interface is designed with two objects for walking and rotating respectively (Figure 8(b)). The second one allows a user to translate and rotate the viewpoint simultaneously with two hands. We also observed that users preferred shorter sliding in the first study. Because of the Lemur's local coordinate system, the smaller size of the Multiball object makes the longer movement with the same length of sliding. Hence, we reduced the object size (i.e. touch area) in the second interface.

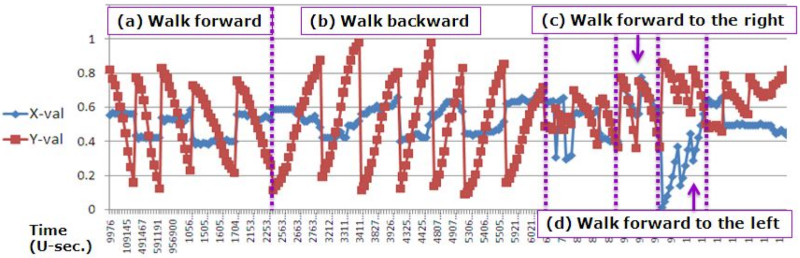
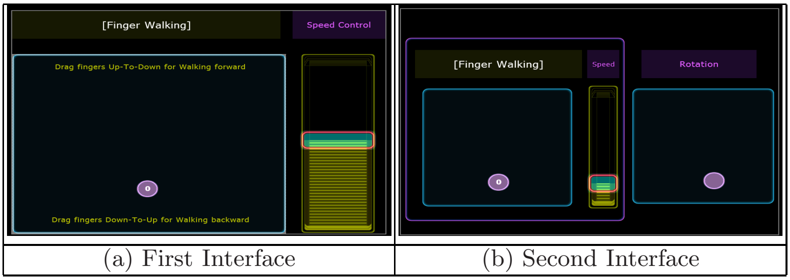


Fig. 7. Finger walking pattern; (a)Walk forward: per sliding, y–value is changed over the 0.4–range from the bigger number to the smaller one. (b)Walk backward: per sliding, y–value is changed over the 0.4–range from the smaller number to the bigger one. X-value is not considered in both (a) and (b) unless x-value is over 0.02–range per sliding. (c)Walk forward to the right: per sliding, both y–value and x–value are changed to the same direction. (d)Walk forward to the left: per sliding, y–value and x–value are reversely changed. The virtual movement direction is determined with the vector of y–value and x–value.



Drag fingers Up-To-Down for Walking forward

Drag fingers Down-To-Up for Walking backward

(a) First Interface

(b) Second Interface

Fig. 8. UI change from (a) walking only to (b) walking and rotation

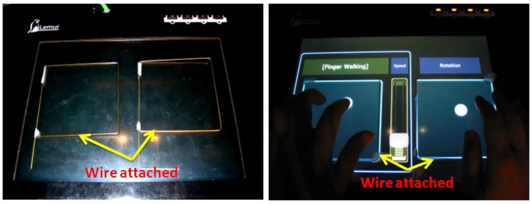


Fig. 9. Wire fit to the Multiball objects on the Lemur

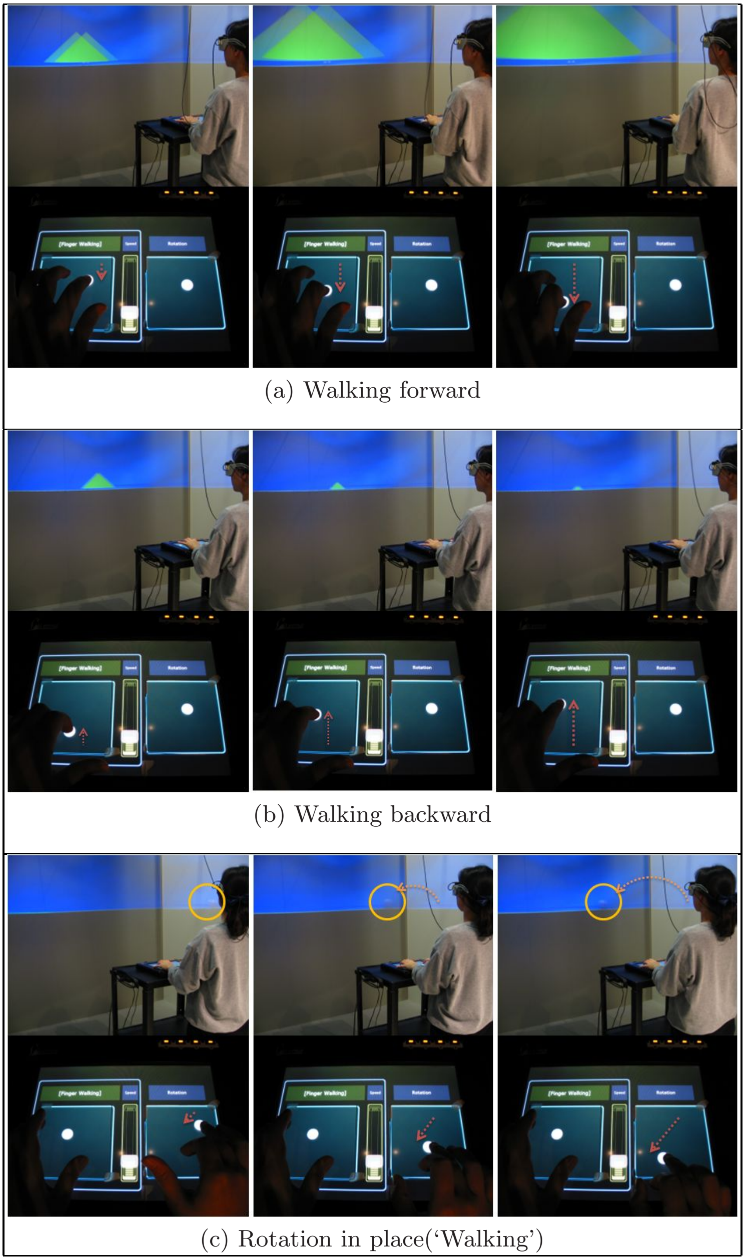
- **Tactile constraint:** In order for users not to touch out of the Multiball object area during navigation, we attached a tactile constraint fit to the walking areas on the Lemur surface. We first used the sticky tape, but later changed to the thin wire (Figure 9) because it is more reusable and easily detached than the sticky tape from the surface. This tactile constraint prevents users from leaving out of the walking areas.

3.3 Formative Study

Twenty five users participated in the second study. Five of those users evaluated our initial implementation redesigned from the first study, and the rest of them evaluated the stabilized interface. The first nine users evaluated the walking–motion rotation technique and the remaining users evaluated the dragging rotation technique. The study procedure was pretty much same as in the first study except for the actual tasks. We asked participants to reach the red cone object, go back to the green cone (i.e. the start point), reach the target(s), and then go back to the green cone again in order. Thus, each task has four sections in total. All participants found the target(s) in seven different tasks; (1) go straight and find one target at 25m, (2) go forward to the right to find one target at 22m, (3) go forward to the left to find one target at 22m, (4) go forward to the right to find one target at 35m, (5) go forward to the left to find one target at 35m, (6) go forward to the right to find two targets at 20m and 35m, and (7) go forward to the left to find two targets at 20m and 35m away from the start point. We took videos of users' motions during the experiment. We focused on usability of FWIP without comparison with other types of techniques. Figure 10 shows user interactions using "walking forward", "walking backward", and "rotation in place" techniques.

Subjective Result and User Feedback. Users showed various but similar finger movements. For long distance, they usually used two fingers of one hand. For short distance, they carefully used one finger-sliding. One user used two hands to walk forward with two fingers, and she moved forward very fast. Some users who play 3D games a lot used both walking and rotation techniques with two hands simultaneously to reach the target. We observed that the tactile constraint attached (refer to Figure 9) on the surface was very helpful for users to reduce the unnecessary context-switching between the surface and the CAVE screens. For the data analysis, we only considered twenty participants since we stabilized the second interface after five users tested it. They evaluated the techniques, walking forward, walking backward and rotation techniques in categories of Satisfaction, Fastness, Easiness and Tiredness using a 1 to 7 scale. For the first two categories, the higher value is better, but for the rest categories, the lower value is better. Figure 11 shows the average ratings as follows in the order of walking forward, walking backward, and rotation techniques: Satisfaction ((5.6), (4.9), (3.7)), Fastness ((5.1), (4.8), (4.3)), Easiness ((2.0), (2.4), (3.5)), Tiredness ((3.1), (2.8), (2.5)).

We used a one-way ANOVA to analyze the subjective results of three techniques in four categories. There was a significant difference between walking forward and rotation in two categories, Satisfaction ($p=0.000 < .05$) and Easiness ($p=0.004 < .05$), respectively. Most of the participants felt very comfortable with the walking forward technique. According to their comments, they gave the higher rate to it because they used a lot this technique to complete the tasks. Compared with walking techniques, rotation techniques were not quite natural to participants. For example, with the 'walking' rotation technique, they



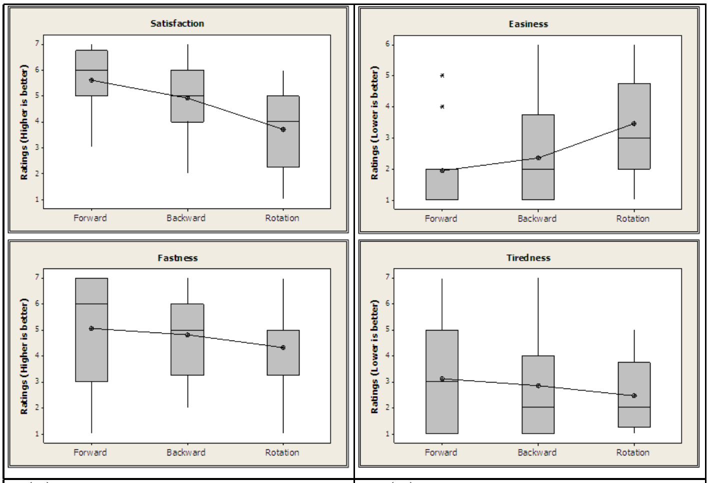
(a) Walking forward

(b) Walking backward

(c) Rotation in place('Walking')

Fig. 10. User Interaction

were not aware of how much they rotated until they got the visual feedback. Sometimes they over-rotated or under-rotated the viewpoint in place than they thought. Since the users did not look at their hands over the surface, in order



(a) Satisfaction and Fastness (b) Easiness and Tiredness

Fig. 11. Subjective Results

to estimate the rotation angle, they might have to rely on how long they are continuously touching the surface, rather than how much angle their hands turn or how much length they drag. They were slightly less tired in 'dragging' (average rate=2) than in 'walking' (average rate=3) for rotation. Based on their demographic information, we can assume that the reason is because 'dragging' technique was quite familiar to the users who play computer games a lot. They also commented that it was not really tiring their fingers. Rather, it was boring to keep on walking by moving fingers. Since the walking backward is very rare even in the real world, they used the walking backward technique only in the case that they already knew the position of the target object behind them.

4 Summary and Future Work

This paper presents FWIP as a novel interaction technique with bare fingers and a multi-touch device for navigation in VEs. FWIP allows users to navigate in VEs as their fingers slide on a multi-touchable surface. They can change the viewpoint by turning the hand, or by using a rotation-in-place technique. Our experiments showed that FWIP can be used as a traveling technique without accurate motion detection. The experiments illustrated our efforts to make the FWIP technique robust as a scaled-down walking-in-place locomotion technique, so that it can be used as a reliable traveling technique.

In future work we will provide the third rotation technique, 'jog-dialing', and compare it with the other two rotation-in-place techniques. If there is no significant difference among three rotation-in-place techniques, we will allow users to use any technique while they are traveling in the virtual world. In order to thoroughly compare FWIP with the most common locomotion technique in VEs, i.e. a joystick-based locomotion technique, we will use a more complex virtual world, such as a maze, where users have to frequently change their viewpoint. Since we used an open plane as a virtual world in the first user study, participants preferred using a joystick due to its continuous control characteristic. For the open plane, the continuous controller would be more appropriate used rather than the discrete controller (i.e. FWIP) because they don't have to frequently change their viewpoint to reach the targets. We will objectively measure the task completion time and the total movement distance to examine which one shows the better performance in the complex virtual world.

References

1. Apple: iPhone (Last accessed, January 2008), `http://www.apple.com/iphone/`
2. Apple: iPod–Touch (Last accessed, January 2008), `http://www.apple.com/ipodtouch/`
3. Bowman, D.A., Kruijff, E., LaViola Jr., J.J., Poupyrev, I.: 3D User Interfaces: Theory and Practice. Addison-Wesley, Reading (2005)
4. Darken, R.P., Cockayne, W.R., Carmein, D.: The omni-directional treadmill: A locomotion device for virtual worlds. In: Proceedings of UIST, pp. 213–221 (1997)
5. Iwata, H., Yoshida, Y.: Path Reproduction Tests Using a Torus Treadmill. Presence 8(6), 587–597 (1999)
6. Jazzmutant: Lemur (Last accessed, January 2008), `http://www.jazzmutant.com/`
7. Kim, J., Gracanin, D., Signh, H.L., Matkovic, K., Juric, J.: A Tangible User Interface System for CAVE Applications. In: The Proc. of IEEE Virtual Reality, pp. 261–264 (2006)
8. Peterson, B., Wells, M., Furness III, T.A., Hunt, E.: The Effects of the Interface on Navigation in Virtual Environments. In: Human Factors and Ergonomics Society 1998 Annual Meeting, pp. 1496–1500 (1998)
9. Slater, M., Usoh, M., Steed, A.: Taking steps: The influence of a walking metaphor on presence in virtual reality. ACM Transactions on Computer Human Interaction (TOCHI) 2(3), 201–219 (1995)
10. Templeman, J.N., Denbrook, P.S., Sibert, L.E.: Virtual locomotion: walking in place through virtual environments. Presence 8(6), 598–617 (1999)
11. Virginia Tech: VT–CAVE (Last accessed, April 2008), `http://www.cave.vt.edu/`

An Empirical Study of Bringing Audience into the Movie

Tao Lin, Akinobu Maejima, and Shigeo Morishima

Waseda University, Tokyo, Japan
{lintao,Akinobu,Shigeo}@mlab.phys.waseda.ac.jp

Abstract. In this paper we first present an audience-participating movie experience *DIM*, in which the photo-realistic 3D virtual actor of audience is constructed by computer graphic technologies, and then evaluate the effects of *DIM* on audience experience using physiological and subjective methods. The empirical results suggest that the participation of virtual actors causes increased subjective sense of presence and engagement, and more intensive emotional responses as compared to traditional movie form; interestingly, there also significantly different physiological responses caused by the participation of virtual actors, objectively indicating the improvement of interaction between audience and movie.

1 Introduction

The last decade has witnessed a growing interest in developing entertainment interfaces and technologies of enhancing interaction and communication between audience and movie. Audience-driven movie such as *interactive drama* is a topic of great currency and has been regarded as the ultimate challenge in this area of digital entertainment [1]. Some significant academic research has focused on developing research prototypes for interactive drama (e.g., [2-4]). These examples of interactive drama have shown great potential in improving interaction between audience and movie, but they do not yet exhibit convincing artistic or entertainment values [1]; in particular, the gap between academic research and practical application in the entertainment industry is still wide.

In addition, how to evaluate entertainment technologies is also an open research challenge in human-computer interaction (HCI) field. Traditional HCI evaluation methods, with their frequent focus on efficiency and task accomplishment, are often inappropriate for evaluating the aesthetic and experiential aspects of entertainment systems. Although subjective evaluation methods such as questionnaires and interviews are also used to evaluate user experience, they have inherent drawbacks. Subjective self-reporting is generalizable, and is a good approach to understanding the attitudes of users; however, subjective responses may not correspond to actual experience [5]. Aware that their answers are being recorded, participants may sometimes respond with what they think the experimenter wishes to hear, without even realizing it. Moreover, entertainment experience, in some sense, is process rather than outcome. Utilizing the subjective evaluation of a single data point to represent an entire condition may wash out the detailed variability of entertainment experience, which is not conducive to analyzing the complex and subtle effects of entertainment technology

A. Butz et al. (Eds.): SG 2008, LNCS 5166, pp. 70–81, 2008.

on player experience. Notably, the increasing availability of physiological sensing offers an opportunity for objectively and quantitatively evaluating user experience. Evidence from physiology shows that physiological measurements (e.g., galvanic skin response, heart rate, blood volume pulse) reflect autonomic nervous system (ANS) activity and can provide key information regarding the intensity and quality of an individual's internal experience [6]; moreover, physiological response is involuntary and thus is difficult to "fake" [6]. We argue that capturing, measuring, and analyzing autonomic nervous system (ANS) activity could provide quantitative and objective access to entertainment experience.

In this paper we first present a new genre of audience-participating movie *DIM,* in which the photo-realistic 3D virtual actors (we call it CG character) of anyone created by computer graphics (CG) techniques are able to participate in a pre-rendered CG movie as its roles. Then we use physiological sensing techniques, along with subjective self-reporting, to evaluate the effects of the participation of CG characters, including audiences and friends' CG characters, on audience experience. The results show that the participation of CG characters is able to cause increased sense of presence, engagement and more intensive emotion response, as compared with the traditional movie without audience's participation.

2 Related Work

Psychophysiology has suggested that physiological response provides key information regarding the intensity and quality of an individual's internal experience and is involuntary and thus is difficult to "fake". Based on physiological knowledge, we believe that capturing, measuring, and analyzing autonomic nervous system (ANS) activity could provide continuous and objective access to the DIM experience, in particular used in concert with other evaluation methods (e.g., self-reporting). In this study, we chose to collect galvanic skin response (GSR) and Electrocardiogram (EKG) signals. Researches have suggested that that GSR is a linear correlate to arousal [7] and reflects both emotional responses as well as cognitive activity [8]. GSR is also used extensively as an indicator of experience in both non-technical domains (see [8] for a compressive review), and technical domains [9, 10]. For example, a recent study suggests that change in skin conductance seems to be able to objectively measure presence in the stressful VR environment [11]. EKG signals can offer the electrical activity of the heart over time. We computed heart rate (HR) from the EKG signals in this study. HR has been used to measure attention [12], presence [11], and to differentiate negative and positive emotions [13].

3 Bring Audience into the Movie

We pre-rendered a 12-mintute CG movie *"Grand Odyssey"* using 3D computer graphics technologies as *DIM*'s "background movie", and it includes 20 CG roles who are able to be acted by CG character of audience. These CG characters were constructed by embedding the 3D photo-realsitic face of audience into the "background" CG role (see Fig. 1). Overall the entertainment system can automatically create CG characters in

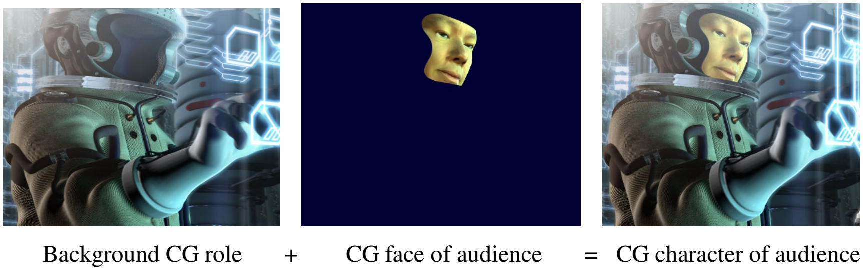


Fig. 1. Construction of audience’s CG character: embedding the highly realistic 3D CG faces of audience into the pre-rendered CG role

several minutes—from capturing the facial information of a participant and generating her/his corresponding CG face, to inserting the CG face into the pre-rendered movie. In the following sections, we introduce several key techniques to create *DIM*.

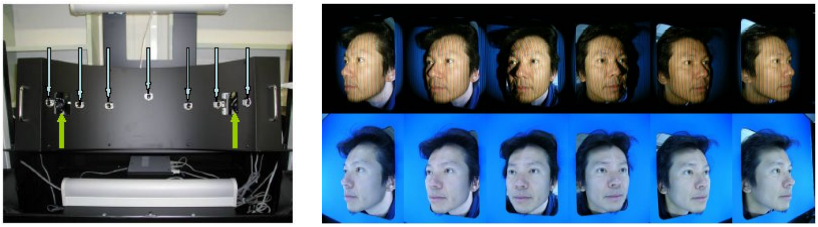
3.1 Acquiring Facial Geometry and Textures

We developed a non-contact, robust and inexpensive 3D range-scanner to generate participant facial geometry and textural information. This scanner consists of seven 800 x 600 CCD digital cameras and two slide projectors which are able to project rainbow stripe patterns (fig. 2 (a)). They are placed in a half circle and can instantly capture face images in few seconds (fig. 2 (b)). The top line in fig.2 (b) is the images with a rainbow stripe pattern and the bottom line is the images taken under fluorescent lighting conditions; and 3D geometry is reconstructed from these images using the hybrid method with an Active-Stereo technique and a Shape-from-Silhouette technique [14]. Fig.3 (b) shows an example of the 3D face geometry constructed by the scanner system. The central camera captures the frontal face image to generate texture mapping (Fig. 3.a).

3.2 Face Modeling

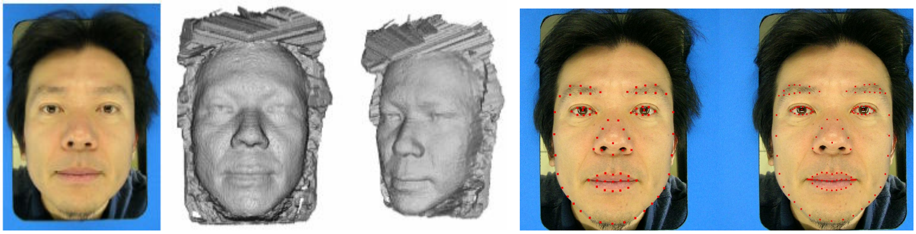
Face modeling is completed in approximately 5 seconds. Overall this process includes four aspects: facial feature point extraction, face fitting, face normalization and gender and age estimation. To generate participant’s personal CG face using captured face texture and 3D facial geometry, it is necessary to extract the feature points of his/her face. We first extract 89 facial feature points, including the outlines of the eyes, eyebrows, nose, lips, and face using the Graph Matching technique with Gabor Wavelet features [15, 16]. Fig. 3 (b) shows the 89 facial feature points which are defined to create a Facial Bunch Graph. We then register Gabor Wavelet features extracted from any facial images (including gender, ethnic characteristics, facial hair, etc) into each node of the Facial Bunch Graph. To enable our feature point detector to correctly extract facial feature points, we collected a total of 1500 facial images as training data.

In the face fitting phase, a generic face model is deformed using the scattered data interpolation method (Radial Basis Functions Transformation, [17-19]) which employs extracted feature points as targets. This technique enables us to deform the model



(a) (b)

Fig. 2. (a): 3D scanner: seven digital cameras (blue arrows) and two slide projectors (green arrows) are placed in a half-circle. (b): The images are captured with six side cameras.



(a) (b) (c) (d)

Fig. 3. (a) texture image, (b) the constructed 3D geometry, (c) 89 feature points and (d) the resampled 128 feature points

smoothly. However, we found that deformed vertices on the model do not always correspond to the outlines of facial areas other than the target points. According to our own experience, it is important that the vertex of the model conform closely to the outline of the eyes and the lip-contact line on the facial image because these locations tend to move a lot when the model performs an action such as speaking or blinking. We therefore must accurately adjust the model for the feature points corresponding to the outline of the eyes, lips, and especially, the lip-contact line. Consequently, we interpolate these areas among the 89 feature points using a cubic spline curve and then resample 128 points on an approximated curve to achieve correspondence between the feature points. This process is performed for eyes, eyebrows, and the outline of the lip. Fig. 3 (d) shows the resampled 128 feature points. There are individual differences in the face size and the face position and angle placing on the scanner window for each participant, whereas the size of the region where the participant's face will be embedded in the background movie is determined by the generic face model. Thus we must normalize all personal face models based on the generic one. We normalize personal faces from the three aspects (i.e., position, rotation and facial size) by [20]. To normalize position, we relocate the rotation center of each personal face model to the barycenter of all vertices from the outline of the eyes because our feature point detector can reliably extract these points. To normalize rotation, we estimate the regression plane from vertices on the personal face mesh and the normal vector of this regression plane can be assumed to be current direction of a face; then we estimate the rotational matrix that makes the angle between the normal vector and the z-directional vector

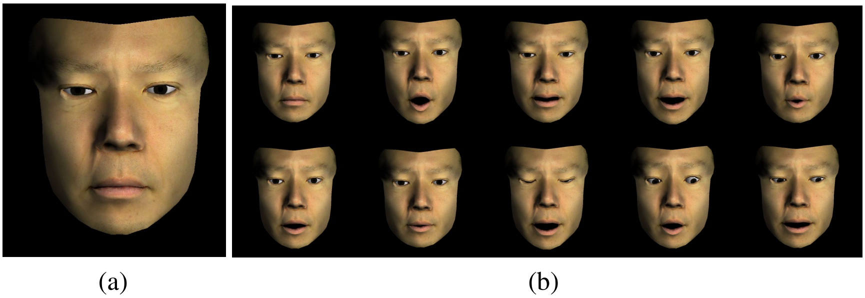


Fig. 4. (a) Personal face model and (b) some examples of the key shapes

approximate zero. To normalize face size, we adjust the size of personal face model, expanding or reducing in size in order to make the ratio between inner corners of the eyes equal to that of the generic face model. This eye ratio is also used to ensure that facial width remains proportionate. Fig. 4 (a) shows the example of personal face model.

3.3 Controlling Character in Real Time

Muscle-based and blend shape techniques are two typical facial expression synthesis approaches. The muscle-based techniques can synthesize high realistic human facial expression, but it is difficult to create target expression using same muscle parameters for an individual, since each person's muscle layout is different; moreover, it is also difficult to simultaneously synthesize facial expressions of several CG actors in real time due to huge computation cost. Although the blend shape-based approach using linear combination of several key-shapes shares the same problem in an individual adaptation, it only causes little computation cost. In addition, we can prepare a variety of basic shape patterns for representing the target facial expression, beforehand. Consequently, we decide to use the blend shape techniques for synthesizing facial expressions.

We first create universal 34 facial expression key-shapes based on "generic face mesh", and then calculate the displacement vectors between the neutral mesh and the specific expressions' mesh (one of 34 key shapes) such as mouth opening. Some examples of key shapes are shown in Fig.4. b. Due to the large individual differences in the width and height of faces for each participant, we must customize the displacement vector. Thus, we introduce a weight to automatically adjust magnitudes for each "generic displacement vector" by calculating the ratios of the face width and height between the individual face mesh and the generic one.

The goal of relighting faces is to generate a matching image between the participant's face and the pre-created movie frames. Considering the requirement for real-time rendering, we implemented a pseudo-specular map for creating facial texture. Specifically, we chose to use only the reflectional characteristic of participants' skin which is acquired from the scanned frontal face images; however, the images still contain a combination of ambient, diffuse and specular refection. We therefore devised a lighting method to ensure a uniform specular reflection for the whole face by installing an alignment of florescent lights fitted with white diffusion filters. To represent skin gloss, we developed a hand-generated specular map on which speculars on the forehead, cheeks, and tip of the nose become more non-uniformly pronounced than other parts of the face. The intensity and sharpness of the speculars were generated by white Gaussian noise. The map can be applied to all participants' faces. Additionally, we approximated subsurface scattering and laungo hair lighting effects by varying intensity on the outlines of the characters' faces.

3.4 Real-Time Movie Rendering

To ensure high performance, reliability, and availability of the whole rendering system, we implemented a cluster system to implement the movie-rendering. The rendering cluster system consists of 8 PCs tasked for image generation of movie frames (Image Generator PC, IGPC) and a PC for sending the image to a projector (Projection PC, PPC)(see Fig. 5. a). The IGPC and PPC are interconnected by gigabit networks. The PPC first receives the control commands, time codes, and participants' CG character information (face model, texture and parameters for facial expressions) from Data Storage Server, and then sends them to each IGPC. The IGPCs render the movie frame images using a time division method and send them back to the PPC. Finally, the PPC receives these frame images and sends them to the projector in synchronization with the time codes. The Fig. 5 (b) shows an example of an audience and his CG character produced by our system in real-time.

In addition, the automatic estimation of gender and age is performed by not-linear Support Vector Machine classifiers which classify Gabor Wavelet Features based on Retina Sampling [21, 22]. The age information from the classifier's output is categorized into five groups: under 10, 10-20, 20-40, 40-60, and over 60. The estimation is used for assigning roles. The entertainment system can automatically select a role from the background movie for each participant according to age and gender estimation, or a role can be assigned manually. In additional, we pre-recorded a male and female voice track for each character, and voices were automatically assigned to characters according to the gender estimation results.

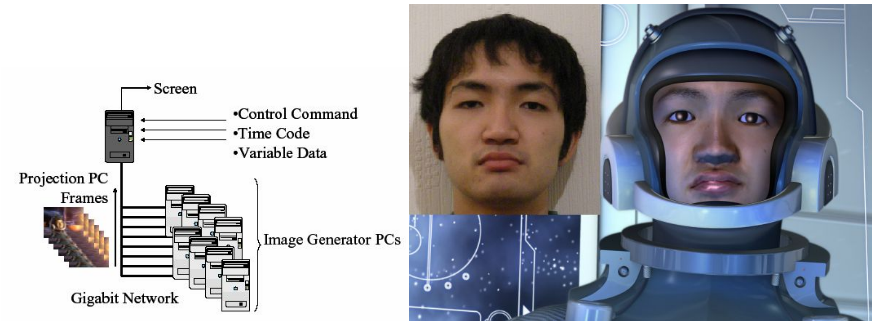


Fig. 5. (a) rendering cluster system and (b) an audience and his CG character

4 Audience Experience Evaluation

In this study we are more interested in the effects of DIM on audience experience, and evaluate the effects using subjective and physiological measures.

4.1 Experimental Tasks and Subjects

To facilitate the investigation of effects, subjects were required to watch three versions of *"Grand Odyssey"* as experimental tasks: (a) *Traditional* version, (b) *SDIM* version, and (c) *SFDIM* version. The order of presentation of experimental tasks is fully counterbalanced. *Traditional* version is like what is normally encountered at the cinema, in which the 20 CG roles were acted by strangers to the subjects. In *SDIM* version, one subject watches her/his own CG character acting a role; and In *SFDIM* version, one subject watches her/his own and 15 friends' CG characters together acting 16 roles in the movie. The 15 people appearing in *SFDIM* version and all subjects were recruited from the same research lab to ensure that they would be familiar with each other and easily recognize the faces appearing in the movie. Twelve university students, 11 male and one female, ages 21 to 24 participated in the experiment. Before the experiment, all subjects filled out a background questionnaire, used to gather information on their movie preferences, experience with the movie and personal statistics such as age and sex.

4.2 Experimental Setting and Protocol

The experiment was conducted in a studio at Waseda University which offers a real theater atmosphere. The movies were viewed on a 150" projection-screen. Since other contextual factors such as resolution, brightness, contrast, sound effects, and temperature could potentially affect users physiologically, we held these factors as constant as possible across the three conditions. Before the experiment, we fitted subjects with physiological sensors, tested the placement of the physiological sensors to ensure that the signals were good, and collected physiological readings during a 5-minute resting period as the baseline for normalizing physiological data, and during the movie-showing time. After completing each task condition, subjects rated their experiences on that task condition using questionnaires, then rested for 4 to 7 minutes. Investigators monitored physiological data during the resting periods to ensure the subjects' physiological responses returned to baseline after rest. At the end of movie viewing, subjects discussed their impressions of the experiment (debriefing).

4.3 Physiological Data Analyses

GSR and EKG signals were gathered using *ProComp Infiniti* hardware and Biograph software from Thought Technologies^{TM}. We measured GSR using surface electrodes sewn in Velcro^{TM} straps placed around two fingers on the same hand, and placed three pre-gelled surface electrodes in the standard configuration of two electrodes on the left arm and one electrode on the right arm to collect EKG signals.

EKG data were collected at 128 Hz, while GSR was collected at 64 Hz. We inspected the HR data and corrected any erroneous samples. There are other factors affecting physiological response, such as sweating, time of day, age, sex, race, temperature, in addition to those factors presented by experimental tasks. Thus, it is difficult to directly compare physiological readings across different task sessions, even for an individual. Physiological response should, then, be regarded as relative rather than absolute. We normalized each physiological signal to a percentage between 0 and 100 within each condition using the following formula:

$$\text{Normalized Signal}\,(i) = \frac{\text{Signal}(i)\text{ - Baseline}}{\text{Signal}_{\text{Max}}\text{ - Singal}_{\text{Min}}}\ \text{x}\,100\ , \qquad (1)$$

where $\text{signal}_{\text{max}}$ and $\text{signal}_{\text{min}}$ refer, respectively, to maximum and minimum values during movie viewing and the rest period; and baseline refers to the average value of physiological data during the rest period. These normalized physiological data reflect physical changes from baseline.

5 Results

5.1 Subjective Response

Audience experience was subjectively evaluated after each condition using the ITC-Sense of Presence Inventory (ITC-SOPI) [23], which identifies four presence-related elements: (a) spatial presence ("a sense of physically being in the movie"); (b) engagement ("a sense of involvement with the narrative unfolding within the movie"); (c) ecological-validity/naturalness ("a sense of the naturalness of the mediated content"); and (d) negative effects ("a measure of the adverse effects of prolonged exposure to the immersive content"). This study employs 35 items addressing only the first three elements. The wording of some items was slightly altered to adapt the instrument specifically to the movie environment. Subjects rated their emotional responses in terms of emotional valence and arousal to each of the conditions using 9-point pictorial scales. Emotional arousal indicates the level of activation, ranging from very excited or energized at one extreme to very calm or sleepy at the other and emotional valence describes the degree to which an affective experience is negative or positive. The valence scale consists of 5 graphical depictions of human faces in expressions ranging from a severe frown (–4, most negative) to a broad smile (+4, most positive). Similarly, for arousal ratings, there are 5 graphical characters varying from a state of low visceral agitation (0) to high visceral agitation (8). Emotional arousal and valence are scored on these two scales, which resemble Lang's Self-Assessment Manikin [24]. Table 1 summarizes the results for subjective rating for experience. A one-way ANOVA showed that there were significant differences in the ratings for spatial presence, emotional arousal, and engagement among the three versions. Also, pairwise comparisons (LSD test) showed that the three movie versions significantly differed from each other: *SFDIM* version elicited the highest subjective ratings for spatial presence, arousal and engagement, followed by *SDIM* and *Traditional* versions, respectively. For emotional valence, there are significant differences across the three versions, but post-hoc pairwise comparisons showed both *SDIM* and *SFDIM* versions caused significantly higher ratings than *Traditional* version and the difference between *SFDIM* and *SDIM* did not reach statistical significance. We did not find any significant difference in the rating for ecological-validity/naturalness.

Table 1. Results of subjectively reported experience. Identifying strongly with that experience state is reflected in a higher mean.

	Traditional	***SDIM***	***SFDIM***	***F***	***p***
Spatial Presence	1.91	2.84	2.90	7.43	0.001
Engagement	2.53	3.33	3.51	6.29	0.005
Naturalness	2.79	2.82	2.72	0.19	0.83
Arousal	2.5	3.78	4.86	7.35	0.02
Valence	-0.03	1.65	2.00	6.33	0.01

5.2 Physiological Responses

We compared the average physiological changes across three task conditions and an ANOVA for normalized GSR shows that there were significant differences in mean GSR changes from baseline among the three task conditions (traditional: 8.47%, SDIM: 2.58%, SFDIM: -5.45%; F (2, 33) = 45.11, p < 0.01); pairwise comparisons (LSD test) also show that mean GSR changes during the three versions differ significantly from each other: the mean GSR changes from baseline during the SFDIM version were greatest, followed by *SDIM* and *Traditional* versions. The "contrast pattern" was consistent for 11 of 12 subjects. Fig.6 shows an example of the raw GSR signals of a subject. For normalized HR, we did not find significant average differences in the three task conditions using one-way ANOVA analysis.

We are especially interested in the orienting physiological response when subjects watch their own and friends' CG characters appearing on the screen (self- and friend-mirroring events), since it is an objective metric indicating the perceptual realism of CG characters. In order to inspect short-term physiological responses to these mirroring events, we synchronized physiological data with the movie scenes and examined small windows of time surrounding the events (a 2.5-second latency was considered). We compared GSR for 5 seconds before and 5 seconds after the events. A subject clearly saw his/her own full-face CG character three times in *SDIM* versions (i.e., 36 self-mirroring events: 3 x 12 subjects). We found that, following a self-mirroring event, subjects tended to exhibit sudden increases in GSR, suggesting an increase in arousal typical of orienting response. A typical example, marked by a red circle, can be seen in Fig.6.b. For all 36 self-mirroring events, subjects exhibited a significantly larger mean GSR increase as compared to those responding to the identical movie scenes in *Traditional* version except that the roles appearing the movie screen were not replaced by subjects' CG characters. (t_{35} = -14.364, p < 0.01).

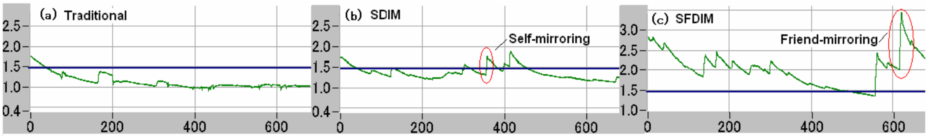


Fig. 6. The example of a subject's raw GSR signals and the blue lines represent the subject's GSR resting baseline. The signals marked by the red circles in (b) and (c) show sudden increases in GSR in response to the appearance of own and friends' CG characters, respectively.

We also examined GSR responses to the friend-mirroring events. We chose *Friend 7* as a typical research target because most subjects reported *Friend 7's* CG character is most realistic and impressive. Fig. 5.b. shows a contrast between friend 7 and his CG character. There were 24 friend-mirroring events for *Friend 7* in all task sessions. Using the same window technology, we found that subjects also tended to experience sudden GSR increases when they watched the friend 7' CG character appearing on the screen, suggesting a sudden increase in arousal. For all 24 friend-mirroring events, subjects experienced a significantly larger mean GSR increase as compared to the GSR changes responding to the identical movie scenes in *traditional* and SDIM versions except that the roles were not replaced by *Friend 7*. (SFDIM vs. SDIM: t_{23}= -10.59, $p < 0.01$; SFDIM vs. traditional: t_{23}= -10.11, $p < 0.01$). We also investigated HR changes for these events and did not find significant differences in HR when watching own and friends' CG characters.

Correlations among measures were performed using the Bivariate Pearson Correlation. Normalized physiological measures first were correlated with the subjectively reported measures across 36 task sessions (12 x 3 conditions). Normalized GSR significantly positively correlated with subjective ratings for presence, emotional arousal and engagement (between GSR and arousal: $r = 0.54$, $p = 0.01$; between GSR and engagement: $r = 0.47$; $p = 0.03$; between GSR and presence: $r = 0.34$; $p = 0.05$).

6 Discussion and Conclusions

Our evaluation shows that the audience-participating movie DIM can elicit stronger subjective sense of presence, engagement and emotional arousal, and increased GSR response. The increased GSR response indicates heighten physiological arousal. Based on previous research findings, we argue that physiological arousal may be a mediating factor. A two-level model of spatial presence [25] has suggested that the focused allocation of attentional resources to the mediated environment contributes to experience of presence. Thus, given that arousal increases attention [26], the increased arousal due to the use of the avatars in SFDIM and SDIM may contribute to increased presence. Another important finding is that the multi-friend participation in *SFDIM* version increases *social presence* ("the sense of being with other body") and causes the larger improvement of audience experience than *SDIM*. For example, in *SFDIM*, the CG character of subject must collaborate with his one of friends to control the spaceship; in the post-interviews, subjects indeed reported the collaboration contributes to the increase of presence. The most interesting result is that subjects experienced oriental responses when they watch their own and friends' CG character appearing on the movie screen, which objectively indicate that audience has "known" that themselves or friends is acting roles in the movie.

However, we did not see significant effects from the two DIM versions on HR; this is inconsistent with GSR. The physiology of the two measures may explain the discrepancy between the GSR and HR results. The cardiac activity resulting in HR is dually innervated by both the SNS and the parasympathetic nervous system (PNS). Increased cardiac sympathetic activity is related to emotional activity and causes the heart to speed up, whereas increased cardiac parasympathetic activity is related to information intake and cognitive engagement, and causes the heart to slow down [27].

We argue that the increased attention and emotional responses may concurrence during the long movie experience presented, and even during a short movie scene, which cause PNS and SNS activity simultaneously. Thus, HR is the resultant effect of PNS and SNS activity and fails to clearly reflect the differences in a single aspect of SNS and PNS activity.

In sum, this study has created the first audience-participating experience DIM, which can enhance interaction and communication between audience and movie; in addition, it also shows great potential of physiological response in evaluating entertainment experience as an objective and quantitative method.

Acknowledgments

This study was supported by the Special Coordination Funds for Promoting Science and Technology of Ministry of Education, Sports, Science and Technology of Japan.

References

1. Szilas, N.: The future of interactive drama. In: The second Australasian conference on Interactive entertainment, Sydney, Australia (2005)
2. Mateas, M., Stern, A.: Facade: An Experiment in Building a Fully-Realized Interactive Drama. In: Game Developers Conference, Game Design track (2003)
3. Cavazza, M., Charles, F., Mead, S.J.: Interacting with virtual characters in interactive storytelling. In: The first international joint conference on Autonomous agents and multiagent systems: part 1, pp. 318–325 (2002)
4. Cheok, A.D., Weihua, W., Yang, X., Prince, S., Wan, F.S., Billinghurst, M., Kato, H.: Interactive theatre experience in embodied+ wearable mixed reality space. In: ISMAR 2002, pp. 59–317 (2002)
5. Marshall, D.C., Rossman, D.G.B.: Designing Qualitative Research. Sage Publications, Thousand Oaks (2006)
6. Andreassi, J.L.: Psychophysiology: Human Behavior and Physiological Response. Lawrence Erlbaum Associates, Mahwah (2000)
7. Lang, P.J.: The emotion probe. American Psychologist, 50, 372–385 (1995)
8. Boucsein, W.: Electrodermal Activity. Plenum Press, New York (1992)
9. Ward, R.D., Marsden, P.H.: Physiological responses to different WEB page designs. International Journal of Human-Computer Studies 59, 199–212 (2003)
10. Mandryk, R.L., Atkins, M.S., Inkpen, K.M.: A continuous and objective evaluation of emotional experience with interactive play environments. In: The SIGCHI conference on Human Factors in computing systems, pp. 1027–1036 (2006)
11. Meehan, M., Insko, B., Whitton, M., Brooks Jr., F.P.: Physiological measures of presence in stressful virtual environments. In: The 29th annual conference on Computer graphics and interactive techniques, pp. 645–652 (2002)
12. Weber, E.J., Van der Molen, M.W., Molenaar, P.C.: Heart rate and sustained attention during childhood: age changes in anticipatory heart rate, primary bradycardia, and respiratory sinus arrhythmia. Psychophysiology 31, 164–174 (1994)
13. Papillo, J.F., Shapiro, D.: The cardiovascular system. In: Tassinary, L.G. (ed.) Principles of psychophysiology: Physical, social, and inferential elements, pp. 456–512. Cambridge University Press, Cambridge (1990)

14. Fujimura, K., Matsumoto, Y., Tetuichi, E.: Multi-camera 3d modeling system to digitize human head and body. In: SPIE 2001 (2001)
15. Kawade, M.: Vision-based Face Understanding Technologies and Applications. In: International Symposium on Micromechatronics and Human Science, Nagoya, Japan (2002)
16. Zhang, L., Ai, H., Tsukiji, S., Lao, S.: A Fast and Robust Automatic Face Alignment System. In: Tenth IEEE International Conference on Computer Vision, Beijin, China (2005)
17. Arad, N., Reisfeld, D.: Image warping using few anchor points and radial functions. Computer Graphics Forum 14, 35–46 (1995)
18. Carr, J.C., Beatson, R.K., Cherrie, J.B., Mitchell, T.J., Fright, W.R., McCallum, B.C., Evans, T.R.: Reconstruction and representation of 3D objects with radial basis functions. In: The 28th annual conference on Computer graphics and interactive techniques, pp. 67–76 (2001)
19. Lee, S., Wolberg, G., Shin, S.Y.: Scattered data interpolation with multilevel B-splines. IEEE Transactions on Visualization and Computer Graphics 3, 228–244 (1997)
20. Hirose, M.: Application for Robotic Version of Range Image Based on Real-time 3D Measurement System, in Chukyo universiy vol. PhD (2002)
21. Yang, M.H., Moghaddam, B.: Gender classification using support vector machines. In: Proceedings of International Conference on Image Processing, 2000, vol. 2 (2000)
22. Lian, H.C., Lu, B.L., Takikawa, E., Hosoi, S.: Gender Recognition Using a Min-Max Modular Support Vector Machine. In: International Conference on Natural Computation Xi'an, China 2005 (2005)
23. Lessiter, J., Freeman, J., Keogh, E., Davidoff, J.: A Cross-Media Presence Questionnaire: The ITC-Sense of Presence Inventory. Presence: Teleoperators & Virtual Environments 10, 282–297 (2001)
24. Lang, P.J.: Behavioral treatment and bio-behavioral assessment: Computer applications. Technology in mental health care delivery systems, 119–137 (1980)
25. Vorderer, P., Wirth, W., Saari, T., Gouveia, F.R., Biocca, F., Joncke, F., Becking, S., Hartmann, T., Klimmt, C., Schramm, H.: Constructing Presence: Towards a two-level model of the formation of Spatial Presence, Unpublished report to the European Community, Project Presence: MEC (IST-2001-37661). Hannover, Munich, Helsinki, Porto, Zurich, (2003).
26. Gatchel, R.J., Gaas, E., King, J.M., McKinney, M.E.: Effects of arousal level and below-zero habituation training on the spontaneous recovery and dishabituation of the orienting response. Physiological Psychology 5, 257–260 (1977)
27. Ravaja, N.: Contributions of Psychophysiology to Media Research: Review and Recommendations. Media Psychology 6, 193–235 (2004)

Creative Sketches Production in Digital Design: A User-Centered Evaluation of a 3D Digital Environment

Anaïs Mayeur[1], Françoise Darses[2], and Pierre Leclercq[3]

[1] LCPC, 58, Bd Lefebvre, 75015 Paris & Ergonomics Laboratory - Paris Descartes University,
45 rue des Saints Pères, 75006 Paris, France
anais.mayeur@lcpc.fr

[2] Ergonomics Laboratory – CNAM, 41 rue Gay Lussac, 75005 Paris, France
francoise.darses@cnam.fr

[3] Lucid Group - University of Liège, 1 chemin des Chevreuils, 4000 Liège, Belgium
pierre.leclercq@ulg.ac.be

Abstract. With their tedious interfaces that impose an encoding sternness, the existing computer assisted architectural design tools (CAAD) too often restrict users' creativity. Paper and pencil are then privileged in the first design stages. In response to this restriction, we have developed a sketch interface, based on an electronic pen and a LCD tablet, which allows the designer to develop ideas and manipulate drawings. This paper describes our observations of five professional architects working with these software and hardware devices. Our analyses are based on session's temporal execution, on graphic production types and on drawing area management. They conclude that the proposed digital solution is compatible with the designers' expectations and really support creative exploration.

Keywords: Sketching, tools for architecture, volumetric representation, spatial cognition, externalization.

1 Introduction

In architectural design, two well-defined stages are observable in the freehand-sketching elaboration: design and production. The first stage, creative in essence, is about exploring shapes and expressions, functional simulations and creative problem-solving. This stage results mainly in uncertainty reduction. The specific properties of the sketching activity are mainly its vagueness, the incompleteness and the ambiguity. Secondly, the production stage aims at specifying the architectural object by defining its parameters, by sketching some non-equivocal and transmissible forms. This phase is allowed by dint of an essentially technical problem-solving and by a structured adjustment of the forms towards a unique model, perfectly described and controlled.

Many Computer Assisted Architectural Design (CAAD) actually exist in the mar-ketplace. Used mainly for quick edition of drawings or models, the digital supports support gives, theoretically, an immediately access at many automatic validations: surface calculating, dimensions control and building's performance evaluation. However

A. Butz et al. (Eds.): SG 2008, LNCS 5166, pp. 82–93, 2008.

under the distinction that we have just mentioned, even if they are powerful tools for *production*, they don't allow to correctly equip the *design* phase. That is probably why the paper-based sketch remains a privileged tool in the very first design phase.

2 The CAAO Tools Doesn't Support Preliminary Design

Two main differences between CAAO tools and freehand-sketches explain the privileged use of sketches in the architect's creative design. Many CAD tools make it possible to design and manipulate digital sketches. However, most of these tools fail at helping the designer in the very first design phases, when the outlines of the project have to be defined and the space of solutions must be explored. This failure is often due to the principles adopted for the user interface, which force architects to implement data that is not relevant when sketching and to choose among a limited set of actions with the mouse. CAAO tools are also based on a univocal representation of the building and are constraining a declarative interaction, i. e. an over-making explicit of implicit elements and a command use predefined.

This rule infers de facto a constrained modus operandi [8] : It requires, in order to protect its unique model that the user adapts itself to it. It guides mechanicaly the human work, sometimes to the extent of narrowing its potential. How could creativity emerge in these conditions? On the contrary, the open approach offered by the sketchbook makes it as a really tool dedicated to the hand, by implementing a direct relation between the mind and the representation. The layout is quick and tolerant: It can also follow the believe flux and allow ideas exploration. The sketch is guided by an economical principle: previous traces are not destroyed and the interrelated sketches' collections keep the image of usable ideas always workable [16].

In the computer sciences, the environment is manipulated some digital objects and some rules of their behaviour. These objects are always predefined and have to be concrete, exact and detailed to answer to the univocity principle. This completeness requires the user's whole attention and thus distracts the user from its primary design task. The expected contents by all informatics system turn out incompatible with the primary design work. On the contrary, the sketch rises with raw, wispy, vague and incomplete layouts. It is a symbolic and polymorphic abstraction, ambiguous and indeterminate and it explores different abstraction levels. The computer science attempts to emulate reality, while the sketch pursuit another objective: to some extent ignore reality in favour of the intention's expression.

To realize a digital drawing or to draw freehand is thus fundamentally two different interactions. But what becomes of these interactions when the sketch is executed with an electronic pen and a digital environment?

3 Digital Surface Use

A lot of researches refer to the user's assessment of the introduction of digital surface in their work activity, even if such technologies don't always manage to compete with traditional media (pen and paper, classic desktop publishing, etc.).

However, few researches exist on the effect of the implementation of such technologies in the domains where the imagination and the creativity are at the root of the activity, as the written work or the architectural design.

An increasing interest for digital surface is identified by their expanding use in more and more varied domains of applications: the hospital environment [21], the school environment and the distance learning [1,3,20], the geological and geographical mapping [4,5], the press [17] and finally the architectural design [6,13]. Beyond these varied domains of application, the digital surfaces support different human activities, which differ from others by their level of creativity.

Some progresses are tangible in the domain of geological and geographical mapping in recent years. Depending of the activity goal, this technology could be implemented on PDA or on TabletPC. Clegg *et al.* [4] have suggested that the use of geological digital mapping implemented on TabletPC is more appropriate for data collection than PDA implementations, which dispose of a more limited capacities processor. However, for more basic data collection, PDA is sometimes preferred for its small size, depending of the geographical site investigated. The inherent metaphor for this technology is the paper sheet on which the user writes some memos and improves it with the edition functions proposed and interoperability (send and reception). The PDA has a very limited surface and processing capacity, that is why it is very often considered as an intermediary device. The data collected on the PDA are thereafter transmitted to better suited devices which offer larger work areas [19].

Other researches are interested by reading and manipulation of digital documents on TabletPC. The "malleable paper" metaphor leads to design reading and manipulation of electronic documents. Wang & Jiang [25] suggested that a digital document implemented on TabletPC allows user to read faster and is preferred, in particular because it allows for faster reading compared to the traditional computer window. Because the activity of reading often goes hand in hand with annotations and highlighting, softwares have been developed which offer these functionalities [24], as well as character and symbol recognition capabilities. Even if the data capture, resulting in a 2D plan, is still carried out mainly by the mouse and the keyboard, the progressive extension of the communication channels between users and systems allow the apparition of digital interfaces in the architects' activity. The multimodal approach in the digital interfaces domain lead actually towards the design of capturing techniques of freehand drawn, so as to recreate sketch's natural conditions by making the designer free of traditional computer tool constrains. Elliott & Hearst [6] compared a large computerized desktop (digital desk) to a standard desktop computer and a small tablet environment for two typical architecture design tasks: sketching and image sorting. The participants' preference was evenly divided between the digital desk and the tablet for the sketching task. This study shows the difficulty of digital environment design for image sorting. According to these authors, to respect architect's practice and activity is the best way to improve the design of such tools.

The sketch assisting tools are less and less perceived as optimized graphical tools but are becoming knowledge-based system integrating artificial intelligence techniques offering the designer new ways of perceiving information and new problem solving strategies [2,9]. The researches which concern these systems take up with interpretation of the layout drawn, by allowing for example the system to detect information's which are not explicitly specified by the user, or to consider the solution in its context [11,15].

The design and the deployment of these new systems, implemented on digital surfaces, are thus aimed at assisting relevantly and in an adaptable manner architects. Regardless, very few researches investigate the impact on the creative activity the use of an electronic pen and digital paper can have.

4 Proposition of a Freehand Design Environment for Preliminary Architectural Design

By using the sketchbook analogy, EsQUIsE-SMA is a design environment made as simple as possible, which can interpret in real time freehand digital drawings. The architect freely draws with an electronic pen on a digital layout. Sketches can be drawn at any place on the layout. Suitable tools and functions for sketching (colors selection, digital eraser, sketch transformations, rooms labeling) are displayed in a menu over the tracing surface. The architect can generate as many layout as s/he wishes and can choose to superpose them or not, in any order. The successive layouts are labeled by the architect and stored in an easy-access pile of drawings.

Once the pile is completed, EsQUIsE-SMA can recognize the drawn components and use them to generate – without human intervention - a 3D model of the building: characteristics which were not explicitly described by the designer are then completed on the basis of an architectural knowledge base. For example, as window heights are rarely specified in the preliminary design, EsQUIsE-SMA automatically selects relevant heights according to the functional type of room.

This 3D model provides the architect with a volumetric view of the artifact (see Figure 1). This view can be oriented in any direction so as to highlight a specific viewpoint. Scale options can also be changed. Other advanced functions (e.g. evaluation of energy needs, topological representation of the architectural spaces) are also provided. We do not describe them here since they are out of the scope of this paper. More information about its technical principles (multiagent system: SMA in French) is available in [10,12].

EsQUIsE-SMA can be used either on a virtual desk (see [22] for further description) or on a Cintiq graphics tablet (see Figure 2). This latter device is the one which we used

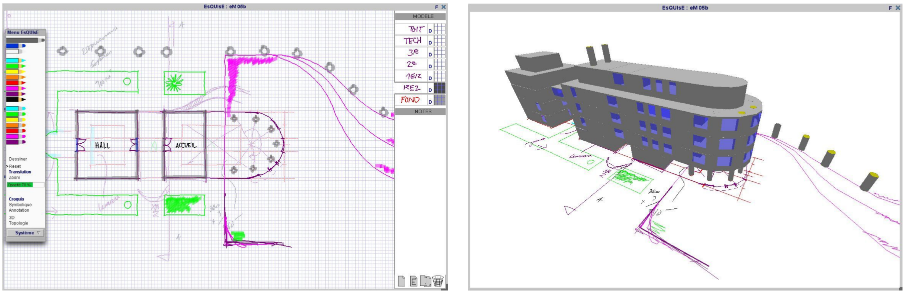


Fig. 1. An overview is drawn on a digital layout with an electronic pen (at the left). Each layout is seen as a floor plan. The architect can then select the layouts which are relevant to him and ask for an automatic generation of the 3D view (at the right).

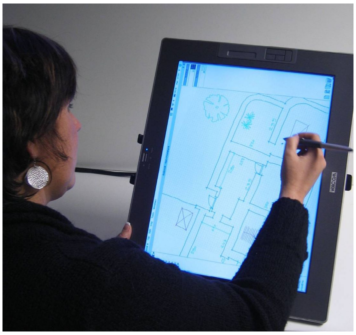


Fig. 2. EsQUIsE-SMA is used on a graphic tablet

in our study. It includes a 21" LCD into the tablet (36 x 29 cm2) which can be rotated or lay flat down so that the user can work as he/she would do on an A3 slate. This technological choice was made because such a device is easy to install and is likely to be adopted by practicing architects.

5 The Study

5.1 Issue: A User-Centered Evaluation of 3D Digital Environment

Considering the previous state of the art, it can be assumed that architects would much appreciate using a CAD tool capable of generating volumetric views of their sketches. The exploration of new ideas would be made easier and their creativity would thus be enhanced. However, the expression of creativity could be impeded by a bad drawing interface. The main purpose of this study is also to evaluate the spontaneous use of the pairs “TabletPC/EsQUIsE-SMA” in preliminary architectural design activity and its adequacy with architects needs in terms of work space.

5.2 Three Successive Steps

The study was divided into three sequential steps. The first one consisted of preliminary interviews (about 30 min.) during which architects were asked about their practices of sketching, and their familiarity with CAD tools. They were also asked to describe how they usually communicate their first blueprints to the customers. The second one was aimed at solving an architectural design problem, using the device EsQUIsE-SMA + Tablet. The third step of the study consisted in evaluating the software, on the basis of a usability questionnaire. This took about 45 minutes.

5.3 Participants: Practicing Architects

As described above, the study aimed at performing a user-centered evaluation with practicing architects. The opportunity of using an attractive freehand digital environment helped a lot in convincing seven architects to give one half day of their working time. Among them, two trial sessions could not be fully recorded for technical

Table 1. Architects' profiles

	Level of experience	**Main job**
Ae1	Expert (31 years)	Co-manager of an agency (5 people)
An2	Novice (1 month)	Unemployed yet
An3	Novice (1 year)	Assistant - architect
Ae4	Expert (40 years)	Works alone in his agency
Ae5	Expert (40 years)	Manager of an agency (2 people)

reasons and are not included in the results. Table 1 summarizes the architect's profiles. It can be noted that two novices were recruited so as to assess the role of experience in using EsQUIsE-SMA.

All of them are used to draw their first sketches by hand on tracing paper. Excepting for the architect *A1* who relies upon his/her assistant to transform the blueprints into CAD files, other architects use commercial CAD tools (such as Autocad®, Archicad®, Arc+® or Architrion®) for the downstream design phases.

5.4 A Realistic Architectural Problem

The problem to be solved has been elaborated in order to enable architects to carry out a first design cycle within two hours. The instructions given to them were: (i) to stop when they feel their blueprint is detailed enough to communicate the key ideas about the envisioned building; (ii) to transmit this blueprint through a sketch *and* at least one volumetric 3D view generated by EsQUIsE-SMA.

The problem situation has been elaborated by an architect and building engineer. Its feasibility was tested before the experiment. It was required to build a high school in a urban area. Other detailed requirements were given on a paper document, such as the size of the plot, the number of pupils to host, some of the functional dependencies (e.g. to have the director's office near to the entrance) and functional requirements (a restaurant, a playground of a specific area, etc).

Architects are asked to use the graphic tablet but they also have some sheets of paper, rulers and rubbers at their disposal.

5.5 Method

Data Collection. Design sessions have been audio and videotaped. A simultaneous verbalization ("think aloud method") was used, which benefits and limits are known [7, 23]. We will not discuss this point here. Verbal recording was not aimed at providing data for verbal protocols but only to support further interpretation of specific sequences. Verbal records were thus not fully transcribed.

Coding Scheme. The first coding scheme was elaborated to highlight the *duration of the drawing tasks*, regarding two factors:

– *Type of drawing* (3D generated view, drawn perspective, 2D overview, cross-section, frontage, text). This latter modality (*Text*) is made up of annotations, calculations, rewriting or interpretation of requirements (note that room labeling is not counted in

this category). A "*Document*" category was dedicated to the reading tasks, during which the architects had to go through the requirements list;
- *Location of the drawing* (main area of the tablet, peripheral area, outside the tablet).

According to this coding scheme, videotapes were segmented into significant units (clips) using iMovie®. The duration of each clip was automatically computed.

Graphical Reports. On the basis of this coding, we have drawn graphical reports which provide a temporal view of each design session. The type of drawing is written in the left column. The clips are numbered. Each color represents one digital layout. Cell borders represent the drawing localization (bold in on the *main area* of the layout, hatched lines are for identifying *peripheral area).* Time duration is written into each cell.

6 Results

First of all, it was necessary to evaluate whether the digital environment, made up of EsQUIsE-SMA and the graphic tablet, would not prevent architects from performing the design process in a satisfactory way.

6.1 Generating Digital Layout

We first wanted to check if the digital device, based on the layout principle, is easy to use. The system must allow the architects to generate as many layouts as necessary to find out their "best" solution. As presented on Table 2, the duration of the design session varies between one hour (*Ae5*) and more than two hours (*An3*). Because of these important disparities, the average time (1h30 min) is not meaningful.

The number of solutions which are explored can be quite low (only two solutions are explored by *An2*) or quite important (up to 9 for *An3*). Since both of these architects are novices, we explain this discrepancy by the style of designing rather than the level of expertise. Except for architect *An2*, the architects generate about 6 different solutions.

It seems that architects do not restrain themselves into a narrow exploration of the solution space when using the digital environment. However, is this exploration easy to perform or not, considering the limited size of the graphical tablet?

Table 2. Heterogeneous practices among architects

	Number of layout generated during the design session	**Number of alternative solutions**	**Duration of the design session**
Ae1	5	6	91'47"
An2	9	2	73'00"
An3	8	9	138'59"
Ae4	12	6	88'50"
Ae5	4	4	67'43"

6.2 Managing the Drawing Areas

The question is whether or not the various external representations needed to accomplish the design process (sketches, annotations, notes, computations) can all be handled on the tablet device itself. If too many of these artifacts are to be dealt with outside the tablet, the usability of the digital environment would have to be questioned. In any case, the sketching tasks which cannot be performed on the device must be identified. It may not be relevant to support all of them, though. For instance, the manipulation of the requirements document, the re-interpretation and re-writing of the constraints may be better processed outside the main drawing space. Regarding the other tasks, additional functions could be required to expand the interface.

Figure 3 provides us with an excellent example of the richness of sketching which is allowed by the device. Architect *Ae1* draws on a sole layout but uses both the central area (2D floor overview) *and* the peripheral area (which is itself divided into two parts: calculations on the top of the area, and a perspective on the bottom).

Results presented in table 3 stress that:

- In most cases, external representations are drawn or written *on the main area* of the tablet.
- Except for *Ae1*, the peripheral area is almost never used. We will report later what it is used for.
- Two groups can easily be distinguished by the duration of the work on each area and especially, by the use of the area "*outside the tablet*" (no statistical test has been done because of the small number of subjects). Contrary to our expectation, these groups do not correspond to the level of expertise, since novices are *An2* and *An3*.

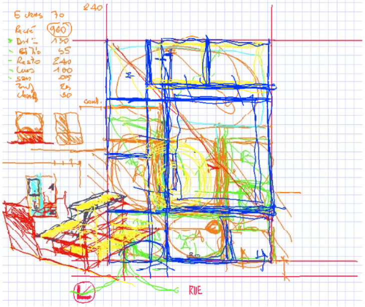


Fig. 3. Diversity of sketches drawn on the different areas of the tablet

Table 3. Duration of work on each area (%)

	Group 1			Group 2	
	Ae1	*An2*	*Ae4*	*An3*	*Ae5*
Main area	82	88,5	88	72,50	62,50
Peripheral area	15	4	6	0	0,5
Outside tablet	3	7,5	6	27,50	37
Total	100 91'47"	100 73'00"	100 88'50"	100 138'59"	100 67'43"

6.3 Writing and Processing Documents

Group 2 spend one third of the design session “outside” the tablet. We have to identify which artifacts are dealt with in this working area. As shown in Table 4 below, there are only two types of external representations which are manipulated outside the tablet:

– Written texts on the available paper sheets (such as annotations, calculations, rewriting the requirements);
– Documents which support the requirements list reading.

There is an important discrepancy between *Group 1* and *Group 2*. These latter architects spend much more time reading the documents and writing texts than the former. We were surprised that professional experience didn't seem to influence this behavior. From observations and interviews, it seems that *Group 2* architects were much more concerned than others with following precisely the requirements given by the experimenter. This could explain why they spend so much time reading these documents, rewriting and interpreting them into additional texts.

This important use of *Texts* by *Group 2* architects do not mean that *Group 1* never writes down a note. But, as highlighted in Table 5, *Group 1* architects do not write so much (5 to 10 times less) and prefer to write down annotations or calculations on the *peripheral area* rather than on paper.

Regarding text writing, the usability of the tablet is thus satisfactory for a classic use. Its peripheral area allows architects to write down usual annotations and calculations. As suggested by some of the architects, a computation function could be added to the interface.

But it’s noteworthy that the peripheral area is only dedicated to writing. Except for *Ae1* who often draws perspectives and cross-sections in the peripheral area (and a few seconds for *Ae4*, for a sole frontage sketch), this area of the tablet is never used to sketch. As a matter of fact, it’s quite surprising that there are so few drawings in this peripheral area, since it is quite easy to access it. We hypothesize that it is a question of work organization style. This interpretation is supported by *Ae1* preliminary interviews

Table 4. Types of external representations manipulated ***outside the tablet*** and duration of work on these representations (min/sec)

External representation	**Group 1**			**Group 2**	
	Ae1	*An2*	*Ae4*	*An3*	*Ae5*
Text	0'00"	0'58"	0’00”	20'20"	9'50"
Doc	3'00"	4'30"	5'30"	17'51"	15'12"

Table 5. Area where ***Texts*** are manipulated and duration of work on these representations (min/sec)

Texts areas	**Group 1**			**Group 2**	
	Ae1	*An2*	*Ae4*	*An3*	*Ae5*
Outside tablet	0'00"	0'58"	0'00"	20'20"	9'50"
Peripheral	3'40"	2'53"	2'32"	0'00"’	0'29"

and post-evaluation. This architect said that he proceeds here as he usually does at his office, by drawing various viewpoints of the artifact on a sole sheet of paper. This provides him simultaneously with different abstraction levels on the same artifact. We will present in section "Results 2" which categories these external representations belong to.

6.4 Would the Architects Use the Digital Environment in Their Daily Practice?

We first assessed the usability of the digital environment for sketching. The results point out that the principle on which the sketching device works is well suited to the architects' needs. The potential of simultaneously sketching on various drawing areas enable architects to keep their own style of designing. The layout principle is flexible enough and easy to use. Our results show that architects do not restrain themselves into a narrow exploration of the solution space. Some of them do not hesitate to generate up to 9 design solutions. Beyond the electronic pen based interface, the observations show that, in terms of interaction process (layouts management and work area management) and in terms of graphical production (sketch content : strokes, symbols, annotations), the tablet seems suitable for supporting architectural design activities.

The usability assessment was done in more details in the third step of this study (see section "The study"). It consisted of a questionnaire (45 minutes long) in which architects had to rate the device (sketching interface, 3D interface and navigation between each interface), according to the conventional criteria of usability [18]: *effectiveness* (can the user carry out its task?), *efficiency* (does the system consume a minimum of resources?), and *satisfaction* (is the system pleasant to use?). These detailed results are not presented in this paper, since they focus on this particular interface and are most likely useful for improving this specific system.

7 Conclusion

The results show that the metaphor on which the digital environment is based is well suited to the architects' needs. The potential of simultaneously sketching on various drawing areas enable architects to keep their own style of drawing. The layout principle is flexible enough and easy to use. No design task carried out outside the tabletPC turned out to be impossible to accomplish within the digital environment. The "TabletPC/EsQUIsE-SMA" desk allows simple and open interactions, personal, vague or ambiguous sketches, yet at the same time offers the computer's ability to evaluate and edit the sketches.

As a conclusion, it appears that some exciting issues are stressed in this study. It opens towards further research regarding the creativity features of digital sketch tools for design. We have to measure how much (and in which ways) the application impacts the design process, and especially, how creativity could be enhanced thanks to such a digital environment.

Acknowledgments. The authors would like to thank the architects who kindly accepted to contribute to this study, partly funded by the French-Belgian research program EGIDE-TOURNESOL, as well as colleagues for their valuable comments on a preliminary version of this paper.

References

1. Anderson, R.J., Hoyer, C., Wolfman, S.A., Anderson, R.: A Study of Digital Ink in Lecture Presentation. In: Proceedings of the CHI 2004 conference on Human factors interaction (2004)
2. Blessing, L.: Engineering design and artificial intelligence: a promising marriage. In: Cross, N., Dorst, K., Roozenburg, N. (eds.) Research in design thinking, pp. 177–212. Delft University Press, Delft, Netherlands (1992)
3. Chang, C.K., Wu, W.Y.: Designing a Novel Input Method for an Asynchronous Forum on Pocket PCs. In: HCI International 2005. Proceedings of the 11th International Conference on Human-Computer Interaction, Las Vegas, Nevada, USA (2005)
4. Clegg, P., Bruciatelli, L., Domingos, F., Jones, R.R., De Donatis, M., Wilson, R.W.: Digital geological mapping with tablet PC and PDA: A comparison. Computers & Geosciences 32(10), 1682–1698 (2006)
5. De Donatis, M., Bruciatelli, L.: Map it: The Gis software for field mapping with tablet PC. Computers & Geosciences 32(5), 673–680 (2006)
6. Elliott, A., Hearst, M.: A Comparison of the Affordances of a Digital Desk and Tablet for Architectural Image Use Tasks. International Journal of Human-Computer Studies 56(2), 173–197 (2002)
7. Ericsson, K.A., Simon, H.A.: Protocol analysis: verbal reports as data. MIT Press, Cambridge (1984)
8. Estevez, D.: Dessin d'architecture et infographie, L'évolution contemporaine des pratiques graphiques. Edition du CNRS, Paris (2001)
9. Jonson, B.: Design ideation: the conceptual sketch in the digital age. Design studies 26, 613–624 (2005)
10. Juchmes, R., Leclercq, P., Azar, S.: A Multi-Agent System for the Interpretation of Architectural Sketches. Computers and Graphics Journal 29(6), 905–915 (2005) (Special issue)
11. Leclercq, P., Juchmes, R.: The Absent Interface in Design Engineering. Artificial Intelligence for Engineering Design, Analysis and Manufacturing, AIEDAM 16(5), 219–227 (2002)
12. Leclercq, P.: Invisible Sketch Interface in Architectural Engineering. In: Lladós, J., Kwon, Y.-B. (eds.) GREC 2003. LNCS, vol. 3088, pp. 353–363. Springer, Heidelberg (2004)
13. Leclercq, P.: Interpretative tool for architectural sketches. In: 1st International Roundtable Conference on Visual and Spatial Reasoning in Design: « Computational and cognitive approaches ». MIT, Cambridge (1999)
14. Leclercq, P.: Programme EsQUIsE. In: Acquisition et interprétation de croquis d'architecture, La lettre de l'IA, n°123, 6e Journées Internationales Interfaces 1997, Man-machine interaction, Montpellier, France, pp. 159–162 (1997)
15. Lipson, H., Shpitalni, M.: Conceptual design and analysis by sketching. Journal of Artificial Intelligence in Design and Manufacturing (AIDAM) 14, 391–401 (2000)
16. McCall, R., Ekaterini, V., Zabel, J.: Conceptual design as hypersketching. In: Proceedings of CAAD Futures 2001. Kluwers, Dordrecht (2001)
17. Molina, A.: European collaboration and the shaping of an interactive multimedia tablet. Technology in Society 21(1), 19–36 (1999)
18. Nielsen, J.: Designing Web Usability. New Riders Publishing (2001)
19. Oomitka, K., Naito, A., Nakagawa, M.: Idea Memo PDA in Scalable Handwriting Interfaces. In: Smith, M.J., Salvendy, G., Koubek, R.J. (eds.) Design of Computing Systems: Social and Ergonomic Considerations, pp. 455–458. Elsevier, Amsterdam (1997)

20. Read, J.C.: The Usability of Digital Ink Technologies for Children and Teenagers. In: Proceedings of HCI 2005, London, pp. 19–35 (2005)
21. Rodriguez, N.J., Borgues, J.A., Acosta, N.: A Study of Text and Numeric Input Modalities on PDAs. In: HCI International 2005. Proceedings of the 11th International Conference on Human-Computer Interaction, Las Vegas, Nevada, USA, July 22-27 (2005)
22. Safin, S., Boulanger, C., Leclercq, P.: A Virtual Desktop for an Augmented Design Process. In: Fischer, X., Coutellier, D. (eds.) Proceedings of Virtual Concept Conference 2005, Research in Interactive Design. Springer, Berlin (2005)
23. Scrivener, S.A.R., Ball, L.J., Tseng, W.: Uncertainty and sketching behaviour. Design Studies 21(5), 465–481 (2000)
24. Shilit, B.N., Price, M.N., Golovchinsky, G.: Digital library information appliances. In: Proceedings Digital Libraries 1998, Pittsburgh, pp. 217–225 (1998)
25. Wang, J., Jiang, C.: Malleable Paper: A User Interface for Reading and Browsing. In: Stephanidis, C., Jacko, J. (eds.) Human-Computer Interaction: Theory and Practice (Part II), pp. 791–795. Lawrence Erlbaum Associates, Mahwah (2003)

Evaluation of an Augmented Photograph-Based Pedestrian Navigation System

Benjamin Walther-Franks and Rainer Malaka

Technologie-Zentrum Informatik (TZI), University of Bremen, Germany

Abstract. Map interfaces are the quasi-standard for car navigation systems, and are usually the first choice for mobile pedestrian navigation systems. Alternatives are being investigated in research and industry that possibly suit the settings and needs of the person on foot better. One solution is augmented reality (AR), which blends navigation instructions with the view of the real world. However, research usually focuses too much on the technical implementation, leaving little time for a thorough assessment of the actual benefits of such a system. In this paper we present an evaluation of a mobile pedestrian navigation system prototype. The system provides a simplified augmented reality experience by presenting visually augmented photographs instead of a real-time video stream. We compare the usability of the AR interface with that of a map-based interface in a field evaluation. Our results challenge the map approach and suggest that AR is not only a serious alternative, but also potentially more suited for route presentation in PNS.

1 Introduction

Navigation solutions are nowadays a well-established aid for drivers, and mobile navigation systems for pedestrians are invading the market. The widespread interface used in these applications is a 2D or 2.5D (tilted plane) map display. The reason for this is that maps have been the predominant means of navigation for centuries, and 2D geographical data is widely available today. However, digital technologies offer far more possibilities than just porting the age-old paper map to a screen. On the contrary, restrictions such as small screen size make a map visualisation possibly less suited for navigation gadgets.

Ideas for optimising or even replacing the map interface have been around for a long time. One of these is augmented reality (AR) – enriching the view of the real world with virtual components. For navigation purposes, augmented information could be additional signs showing the way or giving extra information on the surroundings. Applications for pedestrian navigation, which are more acceptable for experiments as the pedestrian setting is less safety-critical, have been the subject of substantial research.

There is little evidence, however, that such interfaces will be an improvement over the ‘tried-and-tested’ map interface. In fact, research on AR systems focuses very little on such questions, as researchers efforts are occupied to the full by technology. Furthermore, specific techniques for AR visualisation are seldom investigated closely. This could be because functioning real-time mobile AR prototypes are typically too cumbersome and obtrusive to evaluate with actual potential users. Though trends show that

A. Butz et al. (Eds.): SG 2008, LNCS 5166, pp. 94–105, 2008.

hardware is becoming more manageable, this does not let us make realistic observations with the demonstrations available today.

We aim to contribute to current research by testing a low-intrusive mobile navigation system with a non real-time *simplified augmented reality* interface (termed by us sAR in short), which employs the existing concept of augmented photographs [1]. We evaluate this system in field tests in order to assess the potential of sAR for pedestrian navigation by comparing it with a map-based system. The goal is to validate the hypothesis that sAR-based navigation interfaces are an improvement over map-based interfaces for mobile pedestrian navigation, and further to evaluate different techniques of sAR visualisation. Our hope is that insights on a sAR interface can support research on full real-time AR interfaces and applications.

The structure of this paper is as follows: After presenting the state-of-the-art in related fields of research and commercial systems, we describe a mobile sAR pedestrian navigation system (PNS) which we developed as a working prototype. We then show the experimental setup, and analyse and discuss the results of an evaluation of this system.

2 Related Work

Mobile navigation systems for pedestrians is an established area of research. Many systems developed by researchers employ a map display to relay route information [2,3]. Research investigates optimising map presentation for mobile devices by exploring combinations with iconographic or photographic display of landmarks [4,5]. Fröhlich et al. perform an evaluation of four different presentation methods (maps, a radar display, a list of nearby POIs, and augmented reality) [6]. While most research concentrates on outdoor pedestrian navigation, the REAL project [7] developed a resource-adaptive mobile navigation system that uses infrared beacons for indoor position sensing.

Commercial navigation systems for car navigation often feature a pedestrian mode, such as TomTom *Navigator* or Destinator Technologies *Destinator*. As one of the few commercial systems aimed at off-road navigation, *GpsTuner* allows users to add and calibrate their own maps from scans or satellite pictures.

Much research has gone into systems and technologies enabling views of geospatially referenced information in real-time. For example, various projects envision tour guides with inbuilt navigation as an application of mobile AR [7,8,9]. For high-quality tracking these systems commonly utilise sensor fusion of GPS and orientation trackers (inertial sensors, digital compass) as well as optical feature tracking. Because GPS tracking is not available indoors these systems are usually limited to outdoor use. Head-mounted displays (HMDs) are used as display hardware to offer integrated real-virtual vision.

In pursuit of less invasive technologies, researchers have investigated how high-end PDAs can be put to use for augmented reality applications. With the goal of producing a self-contained AR system, the *Studierstube* project implemented real-time feature tracking of fiducial markers on consumer PDAs, while others have even achieved this on a mobile phone [10]. Whilst it has been shown that marker-less tracking is in principle achievable, current approaches are barred from going truly mobile by the limited processing power of handhelds. The Nokia Research project MARA (Mobile Augmented

Reality Applications), achieves (near) real-time video see-through mobile AR on a Nokia mobile phone, providing a first glimpse of what a self-contained real-time AR system could look like [11].

Facing the significant efforts still involved in real-time mobile AR, it is surprising how little work has been done on non-realtime AR techniques. Most notably, Kolbe developed two concepts and prototypes of employing augmented videos and panoramas for pedestrian wayfinding [1]. The first approach shows pre-recorded videos of the whole path to be navigated with overlaid street names, house numbers and information on POIs. In the second approach, sections of 360° panorama photographs are presented to users at every such decision point, depending on their orientation, with *virtual signposts* overlaid on the images giving navigational information.

This brief survey shows that wide-area video see-through AR is only barely possible with current consumer devices. In order to answer questions about how users can really benefit from mobile AR navigation services, we need to make certain compromises.

3 System

Our approach is to shift the focus from technology and implementation to how people use navigation systems. For this we briefly present a system and prototype developed on the basis of existing research on what we call *simple AR* (sAR). The evaluation of this system is then discussed in sections 5 and 6.

Moving away from real-time displays to presenting pre-recorded views in the corresponding context has advantages: As the recording process is unlinked from display and use, position and orientation of the recording device can be precisely determined during recording, as time and technology are not an issue then. The precision required at runtime can be a lot lower, as the final matching of image to real world will be done by the users themselves [1].

Of the two approaches of augmented videos and augmented photographs, the latter seems more suited for experiments on sAR visualisations and interfaces for navigation, as only punctual recording of images is required, yet the full visualisation and interface can still be assessed. For an area model we used the existing method of an attributed graph. Given an origin (often the users current location) and a destination, the shortest route through the network can be calculated. Both area models and route calculation will not be investigated further here. We assume that the route is given as an ordered list of nodes of the graph area model, or waypoints, that the user must pass. At each of these decision points, the user is given a panorama image previously taken at that location. These images form the central part of the sAR navigation interface.

The system is dependent upon a complete set of panorama photographs at decision points of the area to be navigated. These need to be taken as close to the specified position as possible, with fixed camera height and lens settings. For generating spatially registered overlays for urban views it is possible to avoid a 3D city model, and only use an attribute *street width* for perspective projection [1]. However, this approach fails as soon as the topology of the surroundings are needed for precise visualisations. Thus, to properly augment a photograph image with spatially aligned virtual objects, 3D geodata is necessary, including buildings of city geometry. Many companies are producing such models, such as Tele Atlas, and applications such as Google Earth are increasingly incorporating 3D data.

4 Interface Prototype

In order to make observations on the usefulness and usability of incremental guidance with augmented photos, we implemented a prototype for the city of Bremen. The prototype should work on a small section of the Bremen inner city, exemplified by a representative test route.

4.1 Test Route

To ensure that evaluation results would be significant, the following criteria for the test route were developed and the path chosen accordingly:

- **Representativeness:** It should be easily arguable why this route is representative of an application scenario, with a significant start point and destination.
- **Length:** The time to walk the route including stops for orientation should be between 10 and 20 minutes, representing a short walk.
- **Variation and extremes:** Clustered and widely spread waypoints; busy and empty stretches; stretches with easily recognisable as well as insignificant surroundings.

The route selected for our prototype system runs from just outside the old Bremen city walls into the heart of the *Schnoor*, which is a picturesque quarter with small houses dating back to the 15th century, and a main tourist attraction. The length of the route is 950 meters and takes a person familiar with the route 11 minutes to walk with short stops at traffic lights. It has three road crossings, two of which have traffic lights, one tram crossing and one tram/bus junction without car traffic to cross.

An area model for the inner city was modelled manually, taking pedestrian zones and tram tracks into account. The route was also chosen by hand. Both area model and route selection could easily be the product of an algorithm that included POIs and landmarks. The route includes 19 nodes of the pedestrian graph as waypoints along the path. Distances between waypoints vary between 15 and 135 meters (52 meters on average). There are areas of densely set waypoints and far-spread waypoints. The route includes straight stretches and stretches with many subsequent turns. There are stretches along narrow streets and open places, which is important when assessing visualisation of navigation instructions and GPS signal reception.

4.2 Photograph Composition and Interface

We took photographs at waypoints in direction of the route, emulating a section of a panorama photograph. We created and composited the AR overlays in the open source 3D software Blender. As a 3D model of Bremen wasn't available, we created a simple model of the inner city by extruding building outlines traced off a high-detail map. The part of Bremen in question is fairly low-rise and buildings themselves were not to be visualised but rather used for occlusion of 3D shapes on the floor, so this was sufficient for the visualisations desired.

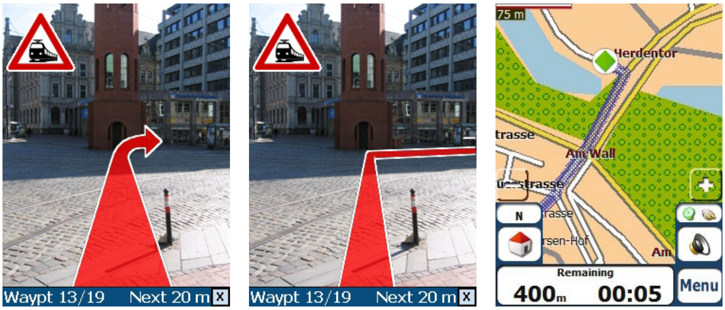


Fig. 1. Screenshots of the prototype interface with arrow and line visualisation; The Destinator 6 interface with a 2D map in pedestrian mode

The physical location of waypoints was mapped to the model and virtual cameras were created at these locations. Using cel-shading techniques we rendered augmented objects in shadeless colours and white outlines on a transparent background. Python scripts then enabled batch rendering and compositing of rendered overlays onto photographs, producing augmented photographs in QVGA screen size.

Existing approaches to AR interfaces for PNS fall into two categories: explicit and implicit route instructions. We chose to incorporate both visualisations:

- **Explicit instructions:** Floor-projected *arrows* point directly toward the next waypoint. In order to prepare for upcoming turns, we indicated them by bending the tip of the arrow in the turning direction.
- **Implicit instructions:** A *line* drawn by connecting waypoint to waypoint exactly depicts the path to walk. To control the line course at sub-waypoint level, we introduced auxiliary nodes.

User observation in first mock-up tests suggested that the attention drawn by the image display is enough to prevent users from noticing potential dangers, such as walking on tram tracks in the pedestrian area. To counteract this and raise awareness of road hazards, a 2D notice modelled after traffic signs was introduced. The same tests showed that an additional orientation indicating how far the user has proceeded along the route and a hint about how far the next waypoint is away is desired in addition to the visualisation.

We included four interface components (figure 1): The photo/image display, an alert symbol (where applicable), which waypoint the current information is associated with (to give an idea of progress along the route) and the direction in meters to the next waypoint. These additional display elements were deliberately treated as non-realtime information to keep in sync with the incremental 'one picture-per-waypoint' interaction. Further modifications included increasing the photo brightness for better visibility on the device screen.

4.3 Prototype

Our evaluation prototype was an offline system with no wireless connection and all necessary data on board. As the data components (images, route description) are uncoupled from the front-end, this can be easily converted into a client/server architecture. Route data was stored as every waypoint signed by ID, their order, exact geographical location (latitude and longitude), radius, and alerts to be displayed. Prototype functionality included receiving and parsing GPS sensor data, constant checking for waypoint proximity and display of augmented image, waypoint number, distance to next waypoint and road traffic notices when within range. In experiments we found a waypoint radius of 12 metres to be a good trade-off between receiving instructions too far off-point and missing instructions due to GPS imprecision. Alerts were implemented so that when receiving new directions the device would vibrate for one second and play a sound. Overall the hardware requirements of the system are pretty low, so it could probably run on low-end devices as long as screen size and quality is sufficient. For our tests we used an Asus P535 PDA phone with integrated SiRFstar III GPS.

5 Experimental Setup

We decided not to determine system usability by setting standard tasks and measuring time of completion, as the effects of a novel interface can make such measurements insignificant. This is because users will tend to spend more time the first time they use an unknown type of interface that provides new visualisations and interactions than with an interface which everybody knows and has used, as is the case with map displays. Rather, we wanted to observe user behaviour and record satisfaction in a standard test and interview and receive information on 1) what the differences were in using the image-based and the map-based navigation system, and 2) how two different types of visualisation – explicit and implicit navigation instructions – compare for the image-based system.

5.1 Test Group Selection

In order to address both the goals of comparison to a map-based system and comparison between different image-based visualisations, participants were divided into three groups: Two using an image-based solution and one acting as a control group using a map-based solution. As a group size we chose five as a trade-off between effort and minimum number for quantitative relevance.

We allocated the 15 volunteers to the three groups based on a self-assessment form querying gender, experience with mobile devices, experience with navigation aids and knowledge of the area. Our goal was to keep the overall profile of the groups similar. Participants were between the age of 24 and 33, which made sure that any variation in results could not be down to different age groups.

5.2 Technical Setup

Participants were handed the PDA phone running the working prototype of the image-based system, or the commercial car navigation software *Destinator 6* running in pedestrian mode (figure 1). Although Destinator has an option in the map orientation settings for ‘forward is up’, this does not change the map orientation from the default ‘north is up’, a fact with significant consequences for the evaluation, as shall be seen later on.

To gain data on system performance and user behaviour in specific situations and to see where problems arise the method of user observation was chosen. Since little to no actual handling of the user interface would need to be recorded, we found simple note-taking sufficient.

For a quantitative measure of usability, the public domain *System Usability Scale* (SUS) [12] as an example of a simple, robust questionnaire used in the industry was employed. It gives a 0-100 scale that is calculated from 10 statements on the systems usability that are graded with a *Likert scale* by test persons. Positive and negative statements are alternated in order to prevent response biases caused by test persons not having to think about each statement.

As a further qualitative insight method an oral interview was prepared. Five questions were composed querying the following aspects of the PNS:

- Usage of **interface elements**
- Perceived **signal precision**
- Ease of matching the **depiction of reality** (map or image) to the real world
- Ease of matching the **navigation instructions** to the real world
- The felt degree of **attention** that the system needed

A last open question was added to include any additional comments that the user still had and weren't covered by previous questions.

5.3 Conducting the Evaluation

All tests took place between 10 a.m. and 3 p.m within a testing time span of three weeks. Test persons were asked to think aloud to facilitate supervisor observation, and to be aware of the traffic. Though there was initial concern that this would overtax test persons, this was soon disproved. In order to simulate the unfamiliarity of the area the route destination was unveiled to participants depending on their degree of knowledge of the test-route area, subjects were only told their destination (the *Schnoor* quarter) if they gave a low score for assessment of how well they know their way around the city. This prevented test persons who know the area well from finding their way to the destination even without help.

No route handling was necessary on the part of the users. We sent users on their way with the device and the supervisor following. As our image-based prototype only covered the route waypoints, system functionality beyond the route had to be emulated. Thus, when test persons deviated too far, the supervisor pointed users back to the closest route waypoint when within radius of a non-route pedestrian graph node. After arriving at the destination, we asked participants to fill out the SUS questionnaire based on their experiences during the test, and subsequently conducted the interview. Test runs took between 30 and 45 minutes altogether.

6 Experimental Results

6.1 Observation

The sensor performance during tests was mixed. The GPS signal showed no correlation to weather conditions. It was unreliable in the narrow streets of the *Schnoor* and better in open spaces, but seemed worst at the corners of open areas. For both systems a low signal quality resulted in irritation for the users. With the map-based system users could judge signal quality more easily as the position icon would move about in an irregular fashion. With the image-based solution people only noticed the low signal quality by absent waypoint instructions, which left them unsure in places. Occasional missed waypoint instructions due to low GPS reception were distributed equally along the route.

Despite problems with the signal users found their way fairly accurately with little deviation from the original route. Incidents of users diverging from the route (walking at least 10 meters unknowingly in a wrong direction) occurred 7 times with the 5 users of the map-based system and 10 times with the 10 users of the image-based system. Of the latter, 6 incidents required the intervention of the supervisor, who redirected them to the nearest route waypoint, the rest found their way back to the route by themselves.

Figure 2 shows where members of different test groups tended to diverge. Map users had trouble at a tram/bus junction (waypoints 28, 26, 27) and near the end of a straight stretch (waypoint 53). Both areas have in common that several street intersections appear close together, and that open spaces are not shown as such but rather as streets along the borders by the map display. Users of the image-based system went wrong

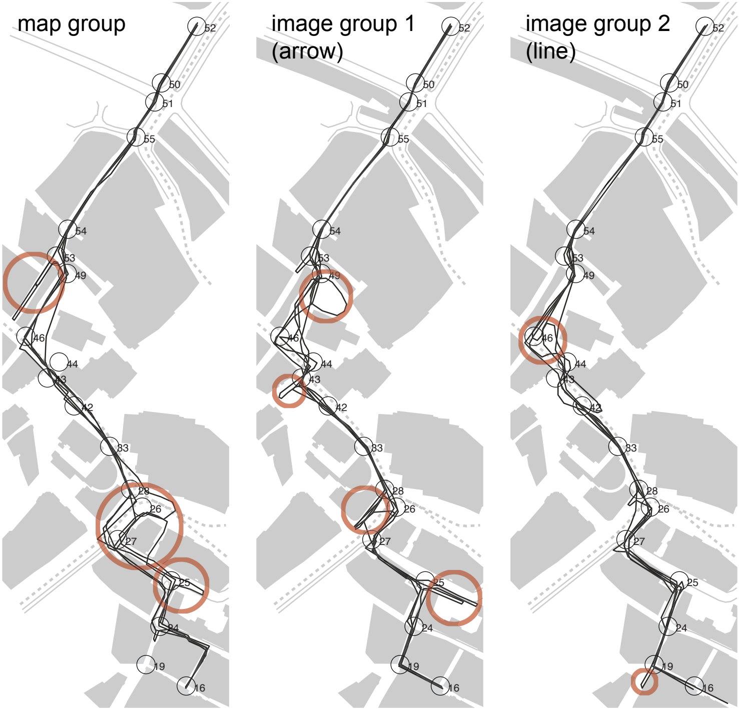


Fig. 2. Paths walked by participants using the map-based system (left) and image-based prototype with arrow (middle) and line visualisation (right). Red circles indicate where participants strayed from the route.

when they could not judge the depth of the turn visualisations at waypoints 44, 28 and 49 (this happened five times with the arrow display as opposed to one for the line display), or when waypoint dropouts occurred due to bad GPS reception. Users of the image-based line visualization strayed least from the route.

There was a constant irritation felt amongst control-group members concerning the north orientation of the map. Every member of that group turned their device so that forward was up in their view. At intersections the map users would stop and consider the direction. Image-based users on the other hand found it easier to orientate, as images pointed the way to go and the overlaid instructions prepared them for turns coming up. On the downside, people were left stranded when no new instructions came due to a low signal reception or misinterpreting previous instructions, making them insecure. The higher temporal resolution of display updates in the map-based system meant that users could check on their current position when they needed. On the other hand, some test persons complained about directions coming in too frequently on straight stretches such as between waypoints 43 to 28.

The traffic symbols employed by the image-based system to give notice of nearby road/rail intersections were recognised and raised the awareness of traffic dangers. Unexpectedly, just as many participants used the traffic signal to disambiguate navigation instructions. In several situations where users were unsure of the distance to walk before turning (notably at waypoints 44 and 28), they saw the symbols as proof that they would have to cross the road/rails and then turn, rather than turn before them.

Even when images were received off-waypoint, users were able to find depicted landmarks or buildings if they were close by, suggesting that the assumption on the allowed difference between viewed and displayed scene is correct.

6.2 Questionnaire

The average SUS scores per group are shown in table 1. While scores for the two image groups are rather similar (86.0 and 85.5), there is an apparent difference to the average score which map group members gave (63.0). Taking the sufficiency of five people per group for granted, one-tailed Student's t-tests show that the average SUS scores of the image groups are statistically significantly higher than the average of the map group, supporting the hypothesis that an sAR interface improves the usability of a PNS over a map-based interface.

When examining average scores per group by statement (table 2), noticeable differences between the map group and the image groups emerged with three statements. Users of the image-based system seemed more interested in frequent usage than users of the map-based system (statement 1). The image-based system garnered less impressions

Table 1. Average SUS score per group. SUS scores have a range of 0 to 100, with a higher value indicating greater overall usability.

Group	0 50 100
map group	63.0
image group 1 (arrow)	86.0
image group 2 (line)	85.5

Table 2. Average ratings of the 10 questionnaire statements per group. Blue bars: map group, yellow bars: image group 1 (arrow), red bars: image group 2 (line).

	Statement	strongly disagree · strongly agree 1 2 3 4 5
1.	I think that I would like to use this system frequently	🟦 2.8 🟨 4.4 🟥 4.2
2.	I found the system unnecessarily complex	🟦 1.8 🟨 1.4 🟥 1.0
3.	I thought the system was easy to use	🟦 3.8 🟨 4.4 🟥 4.6
4.	I think that I would need the support of a technical person to be able to use this system	🟦 1.4 🟨 1.0 🟥 1.4
5.	I found the various functions in this system were well integrated	🟦 3.2 🟨 3.8 🟥 4.2
6.	I thought there was too much inconsistency in this system	🟦 3.4 🟨 1.6 🟥 1.6
7.	I would imagine that most people would learn to use this system very quickly	🟦 3.8 🟨 4.2 🟥 4.6
8.	I found the system very cumbersome to use	🟦 2.6 🟨 1.0 🟥 1.2
9.	I felt very confident using the system	🟦 3.0 🟨 3.6 🟥 3.4
10.	I needed to learn a lot of things before I could get going with this system	🟦 2.2 🟨 1.0 🟥 1.6

of too much inconsistency of the interface (statement 6). Users of the map interface also felt more encumbered by the system than the two image groups (statement 8).

6.3 Interview

When asked about the interface elements they used, it turned out that none of the testers of the map-based system used any interface element apart from the map. Testers of the image-based system mainly used the photographs but mentioned that the notice symbols were helpful in increasing alertness, as well as for disambiguating in wayfinding decisions. The distance to the next waypoint was used sometimes, but users found it hard to judge distances given as numbers correctly. The waypoint information was hardly used at all, which might mean that a better indication of progress along the route is necessary, but could also be due to the fact that participants did not have an actual goal in mind and thus state of progress was not a major concern.

Concerning perceived signal quality, users of the map-based system were more likely to spot signal difficulties, but were then more distracted. Users of the image-based prototype were content when it worked, but noticed that there were dropouts.

Statements on mapping the on-screen representation of surroundings to the real world correspond to observations made during the test run. Map users noted that orientation was difficult. Image users had positive and negative things to say on how well they could recognise images, and stressed the importance of up-to-dateness of the pictures – lack of this led to problems in areas of low permanency such as the market square. Furthermore, users of the image-based system complained about worse recognition in bright light/sunshine, which seemed to annoy map users less. This shows that the image approach is much more reliant on good image quality (lighting, contrast) and visibility of depicted objects. This and the reduced set of information that discrete pictures of parts of the route (namely waypoints) offer make it more fragile than a map-based approach, which offers redundant information.

Regarding support of orientation, test persons of the control group criticised the main problem of rotating the device so that the map was aligned with the real world, and the lack of directions on the screen (both problems were already evident in observation). Orientation with the AR images worked well for close waypoints, but several users named difficulty when arrow/line ends indicating a turn were too small due to perspective distortion. One test person claimed that the depth of the arrow visualization was sometimes hard to judge. This is in accordance with the observation that some users of the arrow-images went wrong at waypoints where arrows indicated a turn.

The amount of attention drawn by the systems was overall judged as medium to low. With both the map-based and the image-based solution users claimed they did not consult the device all the time, but rather only when it was necessary. When asked to elaborate, a difference emerged in that users of the map-based interface could consult on demand, whilst users of the image-based system had to wait for the messages. This fact was rather greeted as positive, as users only felt the need to look at the screen when alerted, and this was usually sufficient for navigation.

7 Conclusion

In summary, it can be said that the image-based system worked well and was well received. Image users were led along the route with little deviation, even slightly less than map users, and had little need for correction by the system. Of the two visualisation types for the image-based system, the line proved to be more useful, as users seemed to be better able to judge the depth of turning instructions. All users found the intended destination, even though it was not revealed. Test persons gave both image-based systems significantly higher usability scores than the map-based system. However, it is arguable that results would have been different with a properly aligned map display, as the difficult orientation was the main complaint of control group test persons. This needs to be investigated in future work.

We have demonstrated that an image-based visual AR approach to navigation system interfaces running on current off-the-shelf hardware works well and copes with the practical issues of the pedestrian environment. Qualitative and quantitative evaluation proved the hypothesis that the image-based interface significantly improves the usability of a PNS to be true. Field tests also showed that AR visualisations using implicit instructions by displaying the path in the surroundings work better for pedestrian navigation than explicit instructions pointing the way. The results challenge the established map approach to navigation system interfaces and show that incremental guidance with augmented images is not only a serious alternative, but also potentially more suited for route presentation in PNS.

References

1. Kolbe, T.H.: Augmented videos and panoramas for pedestrian navigation. In: Gartner, G. (ed.) Proceedings of the 2nd Symposium on Location Based Services and TeleCartography 2004, Vienna, January 28–29 (2004)
2. Aslan, I., Krüger, A.: The bum bag navigator (bbn): An advanced pedestrian navigation system. In: Workshop on Artificial Intelligence in Mobile Systems(AIMS) at UbiComp (2004)
3. Malaka, R., Zipf, A.: Deep map - challenging it research in the framework of a tourist information system. In: Fesenmaier, D., Klein, S., Buhalis, D. (eds.) Information and Communication Technologies in Tourism 2000, Proceedings of ENTER 2000, 7th. International Congress on Tourism and Communications Technologies in Tourism, Barcelona, Spain, pp. 15–27. Springer Computer Science, Wien, New York (2000)
4. Hermann, F., Heidmann, F.: Interactive maps on mobile, location-based systems: Design solutions and usability testing. In: Harris, D., Duffy, V., Smith, M., Stephanidis, C. (eds.) Proceedings of the 10th International Conference on Human-Computer Interaction, vol. 3, pp. 1258–1262 (2003)
5. Beeharee, A.K., Steed, A.: A natural wayfinding - exploiting photos in pedestrian navigation systems. In: MobileHCI 2006: Proceedings of the 8th conference on Human-computer interaction with mobile devices and services, pp. 81–88. ACM Press, New York (2006)
6. Fröhlich, P., Simon, R., Baillie, L., Anegg, H.: Comparing conceptual designs for mobile access to geo-spatial information. In: MobileHCI 2006: Proceedings of the 8th conference on Human-computer interaction with mobile devices and services, pp. 109–112. ACM Press, New York (2006)
7. Baus, J., Krüger, A., Wahlster, W.: A resource-adaptive mobile navigation system. In: IUI 2002: Proceedings of the 7th international conference on Intelligent user interfaces, pp. 15–22. ACM Press, New York (2002)
8. Pyssysalo, T., Repo, T., Turunen, T., Lankila, T., Röning, J.: Cyphone - bringing augmented reality to next generation mobile phones. In: DARE 2000: Proceedings of DARE 2000 on Designing augmented reality environments, pp. 11–21 (2000)
9. Reitmayr, G., Schmalstieg, D.: Collaborative augmented reality for outdoor navigation and information browsing. In: Geowissenschaftliche Mitteilungen (Proc. 2nd Symposium on Location Based Services and TeleCartography), pp. 53–62 (2003)
10. Möhring, M., Lessig, C., Bimber, O.: Video see-through ar on consumer cell-phones. In: ISMAR 2004: Proceedings of the Third IEEE and ACM International Symposium on Mixed and Augmented Reality (ISMAR 2004), Washington, DC, USA, pp. 252–253. IEEE Computer Society, Los Alamitos (2004)
11. Nokia Research Center: Mara – mobile augmented reality applications (accessed May 24, 2007) (2007), http://research.nokia.com/research/projects/mara/
12. Brooke, J.: 21. In: SUS - A quick and dirty usability scale, pp. 189–194. Taylor & Francis, Abington (1996)

Representative Views and Paths for Volume Models

Pere-Pau Vázquez[1], Eva Monclús[2], and Isabel Navazo[1]

[1] Modeling, Visualization, and Graphics Interaction Group
Dep. LSI, Universitat Politècnica de Catalunya (UPC) / Spain
{pere.pau,isabel}@lsi.upc.edu
[2] Institut de Robòtica i Informàtica Industrial, UPC-CSIC
emonclus@iri.upc.edu

Abstract. Volume data models are becoming larger and larger as the capture technology improves. Thus, their visualization requires high computational power. The automatic presentation of volume models through representative images and/or exploration paths becomes more and more useful. Representative views are also useful for document illustration, fast data quality evaluation, or model libraries documentation. Exploration paths are also useful for video demonstrations and previsualization of captured data. In this paper we present a fast, adaptive method for the selection of representative views and the automatic generation of exploration paths for volume models. Our algorithm is based on multi-scale entropy and algorithmic complexity. These views and paths reveal informative parts of a model given a certain transfer function. We show that our method is simple and easy to incorporate in medical visualization tools.

1 Introduction

Optimal selection of viewpoints is an important task in 3D visualization. The amount of information perceived by an observer is determined by the selected view and shading function. Although the visualization techniques are becoming faster and faster, the size of datasets also increases, and the time users are able to devote to a single analysis is limited. Automatic selection of informative views, as well as qualitative exploration paths are useful for book or articles illustration, illustrating models libraries, fast model previsualization, good viewpoints for scene introduction and, in general, as a tool to optimize access to the interesting information and facilitate the understanding of the anatomic structures [1,2]. In this sense, we propose some techniques to help users to find adequate views of the datasets in an efficient way. The presented techniques are useful for general volume models although we mainly use medical examples.

Our proposal works upon a model classified through the definition of a transfer function (or TF, function that maps a density value to an opacity and color value) and the specification of a region of interest. Starting from this minimal information, we generate both a set of representative views of the model and an exploration path that allows users to choose the most suggestive view and, optionally, modify it. The user may also modify the TF if needed, in order to refine the inspection, or to render the same exploration path showing different information. This method is useful, as it allows a user to obtain representative views in a short time, comparable to the loading time of the dataset, and permits the generation of inspection paths at almost no extra cost.

A. Butz et al. (Eds.): SG 2008, LNCS 5166, pp. 106–117, 2008.

In this paper we present an approach for the automatic illustration of volume models via the selection of representative, perceivably different views of volume data, and for the generation of exploration paths around these distinctive parts of the models. The exploration path is built based upon the visually revealed information of the direct volume rendering process, which makes the navigation more closely related to the information that will be perceived by the user. We do not concentrate on TF design, which is orthogonal to our problem. Despite that, since our method emphasizes the information revealed by the TF, we may improve the knowledge of the model by performing the analysis (i. e. selection of representative views, and exploration path) under different TFs. The main contributions of our paper are threefold:

- The selection of a good view criterion that evaluates visual information for volume models. Based on it, we build a fast best view selection algorithm that uses an adaptive scheme. This best view can be chosen as a starting point for the exploration path for on-line inspection, thus reducing the human effort.
- A method based on Kolmogorov complexity that determines the set of visually different views of a volumetric model. These may form the representative set of views of a model, usable to illustrate model libraries or serve as key points for automatic exploration path construction.
- From the information gathered in the previous step, we build an exploration path around the model that reveals a high amount of information from it and passes through the visually different views of the model. This aids the user discovering the most important features of the model with a minimal effort.

The main advantage of our method is that it does not require any preprocess. Moreover, it gives the results in a time comparable to loading a model, and thus, the user can immediately begin the inspection starting from a good view of the model.

Next section introduces previous work on view selection for surface-based and volume-based datasets. Section 3 presents the visual quality evaluation measure we employ and our adaptive best view selection algorithm. In Sect. 4 we develop a method for the selection of representative views based on Kolmogorov complexity. Section 5 gives details on how to build an exploration path around an object that reveals a high quantity of information from it. We discuss our results and summarize our work in Sect. 6.

2 Previous Work

Viewpoint selection has been addressed with different focuses in different areas. In surface-based approaches, the good viewpoints are selected taking into account the geometric information of the models. Usually, three parameters are taken into consideration: i) the number of visible faces, and ii) the area of the object or the visible faces' projections, and iii) the projected silhouette of the object. The analysis is focused under heuristic functions [3] or information theory-based approaches [4]. Several of these methods have been analyzed by Polonsky *et al.* [5] who conclude that none of them can be coined as *universal*. On the contrary, one can always find some object whose best view is *missed* by the developed metrics.

Previous approaches for best view selection of triangular models are not amenable to direct volume rendering because the rendering result is not a set of polygons, and because the usual rendering techniques produce images where, for each pixel, more than a single (iso)value contributes to its final color. Thus, some methods have been developed in order to analyze volumetric information, such as the ones by Bordoloi and Shen [6], Ji and Shen [7], and Takahashi *et al.* [8]. These methods take into account the information contained in the dataset, and thus require a previous preprocess, in some cases quite costly. Furthermore, they do not measure quality based on the rendered views, and thus, different structures might be treated separately although they could produce a uniform colored region on screen. In contrast to these, our approach works on the generated image, with the objective of measuring only the information that will be effectively seen by the user. Mühler *et al.* [1] focus on intervention planning. They preprocess a set of viewpoints placed at 4096 positions on a bounding sphere. At each point, a set of parameter maps is computed that indicate the influence of the current quality parameter settings on the viewpoint. From this information, by using weighted parameter maps that are application dependent, the authors get the information needed to generate cutaway views of the objects of interest. Viola *et al.* [2] have also developed a preprocess method which takes into account, not only a viewpoint quality metric, but also information on focus of attention. Our objective is somewhat complementary to those, as we focus on the fast selection of representative views and exploration paths of models, without the need of segmenting the dataset and minimizing the manual interaction, which is possible thanks to the viewpoint quality measure we selected. However, we may also analyze selected regions of the model as shown in Fig. 7-left.

A closely related problem is the analysis of the information contents in an image. It does not take into account the geometry of the scene but the features of the rendered view. Image analysis techniques have been developed for the automatic selection of lighting configuration from an inverse engineering fashion [9], or with direct image measurement processes [10,11,12]. Those approaches take into account the final image, and measure the information using perceptual or information-theoretic measures. For the view quality evaluation of volume models, we use the approach by [12], presented next, that analyzes the rendered images with a multiresolution entropy measure, due to its simplicity, and the lack of manual configuration required.

3 Fast Best View Selection

As already commented in the previous section we aim at analyzing the information that finally arrives at the user given a certain transfer function. From the metrics we analyzed, the one based on Multi-Scale entropy seems to be adequate for our purposes, as it is simple, easy to evaluate, and requires no user intervention. We will show that this metric, when applied to volumetric data renditions, leads to good views and it is robust to changes in the resolution of the rendered view. Next, we justify our viewpoint quality metric and we explain how to use it for interactive good viewpoint selection.

3.1 Multi Scale Entropy

Typically, the amount of information in an image has been measured using Shannon entropy or one of its derivatives. Shannon entropy gives the average *information* or the *uncertainty* of a random variable:

$$H(X) = -\sum_{i=1}^{N} p_i \log p_i, \tag{1}$$

where $X = \{X_1, X_2, \cdots, X_N\}$ is an image containing integer values and N is the number of different values of a pixel and p_i are the values obtained from the histogram of X, that is, the probabilities of each histogram entry. The logarithms are taken in base 2 and $0 \log 0 = 0$ for continuity; $-\log p_i$ represents the *information* associated with the result X_i. The unit of information is called a *bit*. Observe that this entropy will be 0 if all the pixels have the same value and maximum when all pixels have a different value.

Most of the entropy-based measures for images rely on histogram analysis which, unfortunately, are insensitive to the correlation of pixels. To overcome these problems, Starck *et al.* introduce the concept of multiresolution into the entropy [13] (a.k.a. *multi scale entropy*). They consider that the information in an image is the sum of the information at different resolution levels l. This can be achieved by using a wavelet transform W of an image. Thus, the image information can be measured using the wavelet coefficients $w_{l,k}$ for a fixed set of levels. If the number of levels is high enough, the remaining information can be considered background. The amount of information produced by a rendering can be measured by analyzing the Shannon entropy of wavelet coefficients of each color channel at each level:

$$H_W(X) = -\sum_{l=1}^{L} \sum_{k=0}^{N_l} h_{RGB}(w_{l,k}), \tag{2}$$

where $h_{RGB}(w_{l,k})$ is $-p(w_{(l,k)}) \log p(w_{(l,k)})$, with $p(w_{(l,k)})$ being the relative number of coefficients of the channel with value k in level l. Hence, h_{RGB} means that the entropy is measured separately for each RGB channel. We have applied this method and found that it is suitable for volume rendering analysis because it reasonably analyzes the structures given a TF, as shown next. In [12] a user study showed that the Multi Scale entropy was a good aproximation to the amount of information revealed by illumination. In our implementation we use the Haar wavelet transform, over RGB images encoded in 8 bits per color channel. As shown in [12], four levels of wavelet decomposition is usually enough.

3.2 Interactive Good Viewpoint Selection

The use of the described measure for good viewpoint selection of volume models gives good results, as shown in Fig. 1 and Fig. 2. Here, the quality has been measured for a dense set of viewpoints around a model, and we can observe how it computes higher values (warmer and larger spheres, being the pink sphere the best viewpoint) where more details are provided from the object. Unfortunately, for the purpose of fast model previsualization, a brute-force approach is not practical, because the analysis of a sufficient set of views takes roughly half a minute. As the loading times of the models take several seconds (see Table 1) on a fast machine (a QuadCore PC with 8GB of memory with a GF 8800 card), it seems acceptable to have a good viewpoint selection algorithm that takes at most the loading time. Note (cf. Table 1) that all timings depend on the dataset dimension, number of analyzed views, and the transfer function applied (which

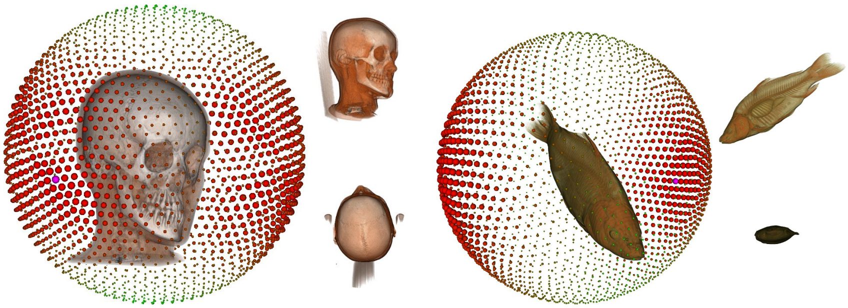


Fig. 1. The quality values for a dense set of views around the head and the fish models. Viewpoint quality is encoded in both the color and the node size (the higher the quality the warmer the color and larger the size). The best view node is painted pink. The images placed on the right of each view quality sphere show the best (top) and worst (bottom) views according to our measure.

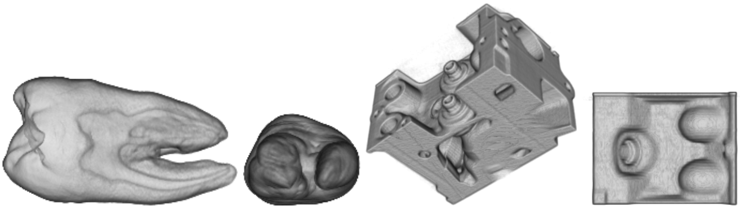


Fig. 2. Best and worst views for the tooth and the engine models

determines the number of processed voxels), that is why a smaller model may have a worse timing than a larger one.

In order to achieve better efficiency than an exhaustive search, we propose two improvements:

Image Resolution: We have analyzed and compared the quality measure for a dense set of viewpoints (2562) on a bounding sphere in order to evaluate how the results depend on the resolution of the viewport. We analyze the images generated offscreen at smaller viewports than the used for regular rendering. Our analysis showed that resolutions of 256×256 are good enough for best view computation for all the models we tested, as best views were identical to the ones selected with 512×512 images. We use 512^2 viewport for high quality rendering, as this size is comparable to the resolution of the models we have. Smaller images (128×128), generally produce good views (very similar if not equal to the optimal) for most models and thus can be used if a quickly response is mandatory (see Fig. 3).

Adaptive Search: In order to achieve interactive rates, we also perform the best view search in an adaptive fashion, starting from a coarse sampling of the bounding sphere and adaptively subdividing it according to an estimator of the entropy.

Our algorithm builds a spherical triangle mesh with the vertices placed at the vertices of an icosahedron. For each vertex i, the quality measure H_i and the total maximum

H_{max} is evaluated and stored. Similarly to Gumhold [11], we treat the quality function as Lipchitz-continuous in a neighborhood of the analyzed points and seek for the Lipschitz constant L of the mesh. This value gives an estimation of the maximum quality variation around a viewpoint.

Our algorithm consists of two steps: First, for each triangle of the current mesh, the maximum reachable entropy point at each edge ($He_1 \cdots He_3$) is estimated using L (see Fig. 4*a*). Then, we estimate the maximum value inside the triangle He_{max} by interpolating the computed values (He_1 to He_3). If $He_{max} + K_{safe}$ is higher than H_{max}, the actual value is measured by rendering a new viewpoint from this position and adding the subdivided triangle (see Fig. 4*a*). We add a safety constant K_{safe} in order to be conservative and avoid missing any maximum. We found experimentally that this constant is only necessary when the range of entropy values of the initial subdivision is not very large, which makes the Lipschitz constant small (below 1). Therefore, we add a $K_{safe} = 0.02 * H_{max}$ when $L < 1$. Each time a new viewpoint is added, the Lipschitz constant is recalculated. Our algorithm stops when none of the estimated entropy values is higher than H_{max}, or when those views are too close (i. e. 5 degrees) to existing analyzed positions. Figure 4*b* shows an example of the subdivision produced for the feet model.

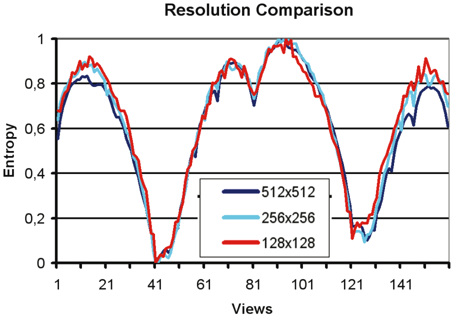


Fig. 3. Viewpoint quality evaluation of the head model under different resolutions: 512 × 512, 256 × 256, and 128 × 128. The camera has covered a circular path around the X axis. Note the high similarity between all the resolutions.

With this approach we obtain very similar maximum values than the ones obtained with the brute-force method, but at a fraction of the time (see Table 1). As an example, we can see in Fig. 4-right two views of the same model, the left one was selected as the best of 2562 views, and the second one with our adaptive method, although the positions are not exactly the same, the information shown is almost identical. Note that,

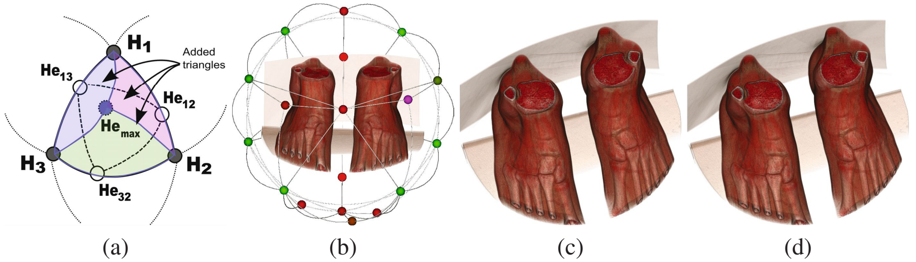


Fig. 4. Left shows the adaptive subdivision algorithm. On the right we can see the result of two steps of the subdivision for the feet model. The best view of the feet model as selected among 2562 views (c) and our adaptive method (d).

Table 1. Comparison of the computation times for different models compared to the loading time. All models have resolution 512 × 512 × *slices*, and *slices* are indicated in column 2. Fourth column shows the seconds required to compute the best view by using a set of 162 positions. The fifth column shows the time needed to obtain the best view with our adaptive method. The last column shows the time needed to find 3 representative views for the model. In contrast to previous columns, where the cost is lineal with size and number of views, here the cost also depends on the compression process which behaves differently for different files of the same size.

Model	**Slices**	**load time (s)**	**Best view** 512^2 / 256^2 / 128^2	**Adaptive** 256^2 **#views / time**	**Adaptive** 128^2 **#views / time**	**Represent. views** 256^2 / 128^2
Head	486	12.09	36.41 / 13.47 / 5.20	28 / 2.32	27 / 0.94	3.85 / 0.40
Torax	291	4.50	25.66 / 8.43 / 3.14	34 / 1.79	26 / 0.56	2.70 / 0.54
Feet	282	4.49	34.2 / 11.13 / 4.03	26 / 1.85	23 / 0.59	1.93 / 0.42
Coxis	100	1.69	18.89 / 5.48 / 2.09	34 / 0.9	18 / 0.26	2.56 / 0.17

for the best view computation, a lower number of regularly placed viewpoints is usually enough, being 162 a good compromise, and therefore this is the number of views we use in the timings shown in Table 1 for the brute force approach.

The time needed to compute the best view using our adaptive method is depicted in Table 1 for different models. Note how, in all cases, we may compute the best view in, roughly, $1/4$th of the time needed to load a model, and with low resolution viewports (i.e. 128^2) even faster.

4 Representative View Selection

When illustrating a complex model, a single view (the one with the highest entropy value) may be insufficient, for example because some details of the object may still be missing or because it does not provide enough information from the 3D relationship of the structures. Selecting views restricting to the ones with high entropy is generally not a good choice because this may lead to show similar information under different directions. This is specially true with models that have a certain level of symmetry. Our brain is very efficient in inferring symmetries, and therefore, it is more valuable to get a set of views that show information not captured in the best view. Although some of them might not be the most informative ones, they will serve as a complement and allow a better understanding of the relationships between the different structures of the model.

Previous approaches require the knowledge of structures present in the dataset, which are obtained via a segmentation process. We would like to avoid this preprocess. In order to do that, we propose an approach that consists in directly comparing the images obtained in the process explained in Sect. 3.1. The view set that adequately represents a model is determined with a greedy scheme:

1. Select the best view B_0 with the adaptive algorithm.
2. Measure the distances from B_0 to the remaining views.
3. Next representer view B_1 is the one at the highest distance from B_0.

The algorithm cost is linear with the number of views, being the comparing process the most costly one. In order to determine the distance between two views ($d(X,Y)$), we use the Normalized Compression Distance as explained in the following sections.

Once we have the two initial representative views, if we want to gather the missed information by these two, we can proceed the same way: We compute the distances from the remaining views against B_1 and choose as new view X the one that maximizes the geometric average of the distances to B_0 and B_1. This process can be repeated several times, but three or four are usually enough for most models. In order to find the optimal set of M views, a global maximization could be applied, but would have a quadratic cost in the number of views and we obtain similar results with our approach.

At this point, the remaining work is to select a robust view similarity measure. Bordoloi and Shen [6] use the Jensen-Shannon divergence over the visible voxels, and do not take advantage of potential symmetry of the dataset, which is desirable. Nonetheless, we cannot compare views by using simple image metrics such as Mean Square Error, as those metrics are sensitive to transformations of the image such as rotation or translation, which may become a problem when comparing two views automatically generated for model inspection. In order to solve these problems, we propose a conceptually simple solution with foundations in algorithmic complexity, concretely, we evaluate view likelihood with the *Normalized Compression Distance*.

4.1 Normalized Compression Distance

Normalized Compression Distance is a universal metric of distance between sequences. It has its roots in Kolmogorov complexity (also known as algorithmic complexity). We briefly detail here some concepts of algorithmic complexity. The interested reader can refer to Li and Vitányi's book [14] for a deeper and more theoretical introduction.

The **Kolmogorov complexity** of a file is the ultimate compressed version of the file. Formally, *Kolmogorov complexity* ($K(x)$) of a string x is the length of the shortest binary program to compute x on a universal computer (such as a universal Turing Machine). Thus, $K(x)$ denotes the number of bits of information from which x can be computationally retrieved. Hence, $K(x)$ is the lower-bound of what a real-world compressor can possibly achieve.

The **conditional Kolmogorov complexity** $K(x|y)$ of x relative to y is the length of a shortest program to compute x if y is provided as an auxiliary input. Both Kolmogorov complexity and conditional Kolmogorov complexity are machine independent up to an additive constant.

Given these two definitions, Bennet *et al.* [15] define the **information distance** between two, not necessarily equal length binary strings as the length of the shortest program that can transform either string into the other one, both ways. The information distance is a metric. Li *et al.* [16] present a normalized version of information distance dubbed *similarity metric*, defined as:

$$d(x,y) = \frac{max\{K(y|x),K(x|y)\}}{max\{K(y),K(y)\}} \tag{3}$$

The authors also prove that this metric is universal (two files of whatever type similar with respect to a certain metric are also similar with respect to the similarity metric).

Being Kolmogorov complexity not computable, it may be approximated with the use of a real-world compressor, leading to the **Normalized Compression Distance (NCD)**:

$$NCD(x,y) = \frac{C(xy) - min\{C(x),C(y)\}}{max\{C(x),C(y)\}} \tag{4}$$

where function $C(F)$ is the size of the compression of a certain file F, and xy is the concatenation of files x and y. Although the similarity metric has values in $[0..1]$, NCD values are usually in the range of $[0..1.1]$, due to compressor imperfections. NCD has been used for applications such as language classification and handwriting recognition [17].

4.2 NCD-Based Representative View Selection

Cilibrasi and Vitányi [17] analyze the conditions that compressors must fulfill in order to be used for computing the Normalized compression distance. They state that most of them do, such as stream-based (*zlib*), block-based (*bzip*), and statistical (*PPMZ*) compressors. As studied by Cebrián *et al.* [18], the use of compressor for comparison purposes requires some issues to be taken into account, for instance, the size of the compressed files limits the efficiency of the comparison processes. In the case of *bzip2*, the *best* option works properly for files up to 900KB before being compressed. Larger sizes make the comparison processes less effective. Based on this, we use *bzip2* compressor on PPM images. Note that the images we analyze are of 256 × 256 or 128 × 128, and therefore, the size of the concatenated file never exceeds 900KB. One might wonder if specialized compressors (i.e. JPEG or similar) would do a better job than general compressors, but, we tested lossless compression using *JPEG2000*, which generated larger files (rougly 20%) than *bzip2*.

In consequence, to determine the distance between two images ($d(X,Y)$), we concatenate them, and then we compress the original and the concatenated files. Then, the distance is measured using (4). In Fig. 5 we show, from a set of views around the head model, the four pairs whose distance is the smallest. Note how symmetric views are correctly ranked as being very similar. An example of selected representative views is shown in Fig. 6, where the three selected views for the head and fish models are shown.

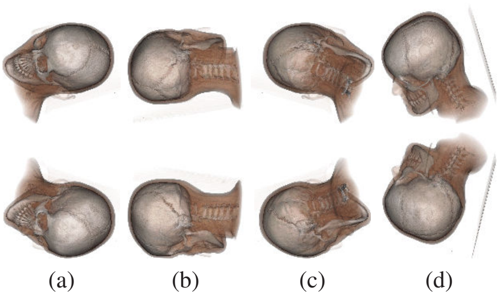


Fig. 5. From left to right, each column shows the pair of images whose distance is the smallest with respect to the rest. Note how our similarity measure is robust with respect to rotation and symmetry. Images are measured as is, that is, without any rotation to align them.

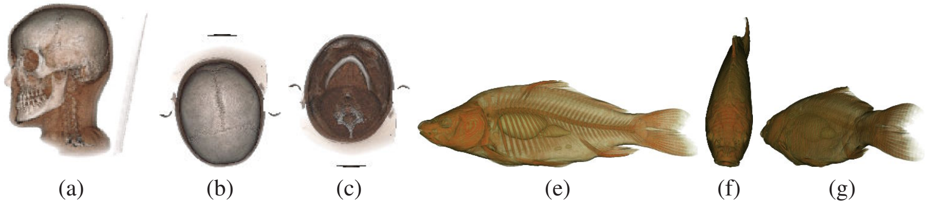


Fig. 6. The 3 representative views of the head and the fish model. Note that they properly represent different appearances of the models.

Observe that these views show substantially visually different information. Representative views computation is quite fast (see Table 1), concatenation and compression are the most costly parts. However, in most cases we still require a total time (adaptive best view selection + representative view selection) smaller than the loading time. Moreover, we may perform the whole process in roughly one second for most of the models tested if we restrict ourselves to offscreen-viewports of 128^2. Because we measure *NCD* over rendered views, our representative view selection method yields views that *look different*. For most models, this will be a key issue, as selecting views according to the visible voxels does not guarantee that the final rendered information will enhance our knowledge of the model (it may be simmetric).

5 Exploration Path Construction

For complex objects, the construction of an inspection walkthrough may be very useful, as a set of views may be insufficient for providing a global vision of the model. Therefore, we propose an algorithm for generating exploration paths for model inspection. This path consists on a set of positions located on a bounding sphere of the object or region of interest. We use as key points the important viewpoints selected in the previous section and ensure visiting all of them. The viewpoint with the highest entropy is visited in second place, as this allows the camera to visit the surroundings of this point that, intuitively, should be more informative than the rest.

The path construction algorithm determines the minimal length path that passes through all the representative views and ensures we are providing a high amount of information. As we have only three to four candidates, an exhaustive search of the shortest path that ensures we pass through the best view in second position is computed instantaneously. We call this the *simple exploration* algorithm.

In order to maximize the information gathered through the exploration, we introduce an *improvement*: the camera may deviate from the short path up to a certain distance at each step. For each step, we analyze the entropy of three candidates, which are placed toward the destination point and separated by ±4 to 6 degrees. The one with the highest entropy is chosen. The allowed total deviation is limited and reduced as long as we get closer to the next key point. This ensures we are visiting all keypoints. Furthermore, the speed of the camera is reduced when visiting points nearby the best one (B_0) because, intuitively, they will show a higher amount of valuable information. In Fig. 7 we can see the results with the *simple path* and the *improved path* for the exploration of the kidney

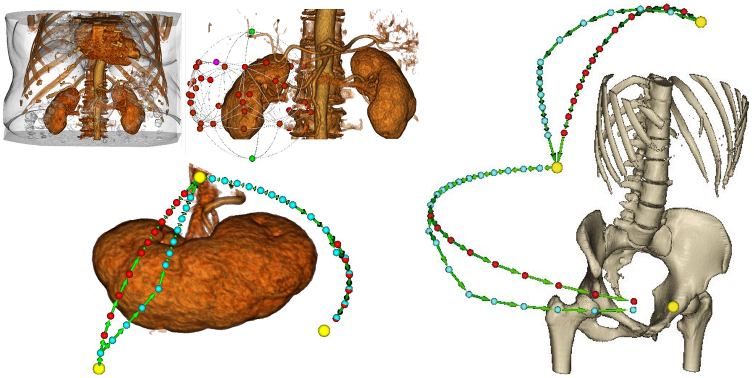


Fig. 7. Left:Analysis of a region of interest around the kidney. Right: A comparison of the simple exploration path (red) and the improved version (cyan) for two models, the kidney and the torax.

and torax models. The improved approach gathers a higher amount of details than the simple method. The *simple path* calculation time is negligible, but the *improved path* requires the evaluation of entropy at each step, and therefore, the time will depend on the number of steps we want to produce.

6 Conclusions

Throughout the paper we have analyzed and commented upon the efficiency and quality of the results obtained applying our techniques either for the good view selection and the exploration path construction. Its fast response, ease of use, and lack of parameter definition, facilitates its incorporation in daily work by i.e radiologists at almost null cost. Nonetheless, its simplicity makes it easy to incorporate in a medical volume visualization package but may also be applied to other rendering metaphors.

We may also apply our technique to a region of interest defined by the user, who also defines the proper TF, and then, our algorithm computes an exploration path around the region of interest. The result is shown in Fig. 7-left. Once the exploration path is obtained, we can modify the TF in order to emphasize different anatomic structures and help in understanding their spatial relationship (see Fig. 7).

In this paper we have explored different techniques for fast volume models exploration. We used an image information contents evaluation based on multi-scale entropy in order to determine good viewpoints for volume models using a fast adaptive method. We have developed algorithms for the selection of representative views of models and the generation of exploration paths. The key difference of our approach is that we use a metric on the generated views, which allows us to abstract from the transfer function or rendering method used. Similarity between views is evaluated by using the notion of *Normalized Compression Distance*, borrowed from Kolmogorov complexity field. Our method is useful, not only for volume model inspection, but also easy to incorporate to medical visualization tools, document illustration support, and automatic labeling of model libraries. One of the main advantages is that we do not require any preprocess, and therefore, our method is adequate for fast previsualization of models.

Acknowledgments

This work was supported by TIN2007-67982-C02-01 of the Spanish Government. Thanks to "The Volume Library" at http://www9.informatik.uni-erlangen.de/External/vollib/ for providing several datasets.

References

1. Mühler, K., Neugebauer, M., Tietjen, C., Preim, B.: Viewpoint selection for intervention planning. In: EG/ IEEE-VGTC Symposium on Visualization, pp. 267–274 (2007)
2. Viola, I., Feixas, M., Sbert, M., Gröller, M.E.: Importance-driven focus of attention. IEEE Transactions on Visualization and Computer Graphics 12(5), 933–940 (2006)
3. Plemenos, D., Benayada, M.: Intelligent display in scene modeling. new techniques to automatically compute good views. In: Proc. International Conference GRAPHICON 1996 (1996)
4. Vázquez, P.P., Feixas, M., Sbert, M., Heidrich, W.: Viewpoint selection using viewpoint entropy. In: Proceedings of the Vision Modeling and Visualization Conference (VMV-01), Stuttgart, pp. 273–280 (2001)
5. Polonsky, O., Patanè, G., Biasotti, S., Gotsman, C., Spagnuolo, M.: What's in an image? The Visual Computer 21(8-10), 840–847 (2005)
6. Bordoloi, U., Shen, H.W.: View selection for volume rendering. IEEE Visualization, 487–494 (2005)
7. Ji, G., Shen, H.W.: Dynamic view selection for time-varying volumes. IEEE Transactions on Visualization and Computer Graphics 12(5), 1109–1116 (2006)
8. Takahashi, S., Fujishiro, I., Takeshima, Y., Nishita, T.: A feature-driven approach to locating optimal viewpoints for volume visualization. IEEE Visualization, 495–502 (2005)
9. Patow, G., Pueyo, X.: A survey on inverse rendering problems. Computer Graphics Forum 22(4), 663–687 (2003)
10. Shacked, R., Lischinski, D.: Automatic lighting design using a perceptual quality metric. Computer Graphics Forum (Proceedings of Eurographics 2001) 20(3), C–215–226 (2001)
11. Gumhold, S.: Maximum entropy light source placement. In: Proc. of the Visualization 2002 Conference, pp. 275–282. IEEE Computer Society Press, Los Alamitos (2002)
12. Vázquez, P.: Automatic light source placement for maximum illumination information recovery. Computer Graphics Forum 26(2), 143–156 (2007)
13. Starck, J., Murtagh, F., Pirenne, B., Albrecht, M.: Astronomical image compression based on noise suppression. Publications of the Astronomical Society of the Pacific 108, 446–455 (1998)
14. Li, M., Vitanyi, P.M.: An Introduction to Kolmogorov Complexity and Its Applications. Springer, Berlin (1993)
15. Bennett, C., Gacs, P., Li, M., Vitanyi, P., Zurek, W.: Information distance. IEEETIT: IEEE Transactions on Information Theory 44 (1998)
16. Li, M., Chen, X., Li, X., Ma, B., Vitanyi, P.: The similarity metric. IEEE Transactions Informmation Theory 50(12), 3250–3264 (2004)
17. Cilibrasi, R., Vitanyi, P.: Clustering by compression. IEEE Trans. Information Theory 51(4), 1523–1545 (2005)
18. Cebrián, M., Alfonseca, M., Ortega, A.: The normalized compression distance is resistant to noise. IEEE Transactions on Information Theory 53(5), 1895–1900 (2007)

Real-Time Camera Planning for Navigation in Virtual Environments

Tsai-Yen Li and Chung-Chiang Cheng

Computer Science Department, National Chengchi University,
64, Sec. 2, Zhi-nan Rd., Wenshan, Taipei 116, Taiwan, R.O.C.
{li,g9401}@cs.nccu.edu.tw

Abstract. In this work, we have developed a real-time camera control module for navigation in virtual environments. With this module, the tracking motion of a third-person camera can be generated automatically to allow a user to focus on the control of an avatar. The core of this module consists of a motion planner that uses the probabilistic roadmap method and a lazy update strategy to generate the motion of the camera, possibly with necessary intercuts. A dynamic roadmap specified relative to the avatar is updated in real time within a time budget to account for occlusions in every frame of the control loop. In addition, the planner also allows a user to specify preferences on how the tracking motion is generated. We will use several examples to demonstrate the effectiveness of this real-time camera planning system.

Keywords: Real-Time Camera Planning, Probabilistic Roadmap, Intelligent Camera Control, Budget-based Planning.

1 Introduction

Real-time 3D games are becoming a popular application of virtual environments in our daily life. Through graphics rendering and interactions, a user can immerse into realistic virtual scenes that are difficult to experience in the real life. In a typical virtual environment such as World of Warcraft and Second Life, a user uses control devices such as keyboard and mouse to navigate and interact with objects or other users in a virtual world. In these virtual environments, a user usually expects to have the control of how to navigate through the environment but may not want to be overwhelmed by the complexity of cumbersome controls provided by the limited devices available on a desktop computer. The quality of a navigation experience usually is highly affected by how the camera is controlled. Thus, how to design an intelligent mechanism for camera control that can ease the burden of the user has been a research topic that has attracted much attention in the past years [10].

Typically, there are two types of camera control that are commonly adopted by 3D games: *first-person view* and *third-person view*. In a first-person control mode, the camera is attached to the eye of the avatar or some short distance behind the avatar. Therefore, moving the avatar would automatically transport the attached camera. However, the view is always limited to the front of the avatar, which may not be

A. Butz et al. (Eds.): SG 2008, LNCS 5166, pp. 118–129, 2008.

desirable in some applications. On the other hand, the control from the third person view detaches the camera from the avatar and provides a more flexible way to view the avatar as well as the surrounding environment. The camera in this control mode can track the avatar from a distance or perform intercuts when appropriate. Although this kind of control is more flexible, it also presents an extra burden for the user to take care of the control of the camera in addition to the avatar. Therefore, it is highly desirable for the computer to take care of the control of the camera such that the user can concentrate on the control of avatar. There has been much research on designing camera control module in a game [11][15] but few can plan camera motions in real time based on the predicted motion of an avatar.

In this work, we focus on the problem of providing an effective third-person control of the camera. We aim to design a real-time motion planner for camera to track the predicted motion of the avatar. The generated camera motion should avoid occlusions from obstacles and follow cinematographic idioms [1] as much as possible. With this automatic camera control module, we hope that the user can concentrate on controlling the motion of the avatar while maintaining a high-quality view. After reviewing related work in the next section, we will describe how we model the planning problem and why we choose a probabilistic roadmap approach in Section 3. In Section 4, we will describe how we modify the planner to take intercuts into account. Then, we will demonstrate the effectiveness of our planner by the several examples from our experiments. We will then conclude the paper with future extensions.

2 Related Work

There has been a significant amount of research into virtual camera control in recent years. According to how the problem is modeled and solved, we can roughly classify them into three categories: *algebraic system*, *constraint satisfaction*, and *planning*.

Algebraic System: In this type of systems, camera control is formulated as an algebraic problem whose solution is the desired camera position [5]. With this approach, the solution can be computed quickly but may lack flexibility. In [9], the authors have developed a high-level language called Declarative Camera Control Language (DCCL) to describe various cinematographic idioms. A compiler has also been developed to convert the camera specifications into parameters for an algebraic system to solve for the camera position.

Constraint Satisfaction System: This type of systems allows the users to specify their preferences on how to position the camera as constraints [2]. Then the system models it as a constraint satisfactory problem and tries to solve for a solution or a set of possible solutions [4][8]. Some systems further model it as an optimization problem according to some criteria to search for a good solution from the set of feasible ones [3][11]. Due to the computational complexity, most of these systems are not suitable for real-time interactive systems such as games. In [11], the authors use an efficient occlusion computation technique to maintain frame coherence in real time. Our work differs from theirs in that we have used a roadmap-based approach to search for a coherent trajectory and adopted the concepts of time budget to ensure real-time performance.

Motion Planning System: This type of approaches was originated from Robotics for the generation of collision-free motions for robots. A good survey of algorithms for this problem can be found in Latombe's book [13]. Due to the computational complexity of this problem, most of the existing planning algorithms were not designed for real-time applications. Nevertheless, some attempts, especially for graphics applications, have been made to generate the motions for digital actors or cameras on the fly [16][17][18]. Like our approach, [16][18] also used the Probabilistic Roadmap Method (PRM) to compute camera or character motions. A probabilistic roadmap usually is built in a precomputation step to make the search in the query step efficient. However, due to the long precomputation time, this approach usually only works for static scenes.

3 Real-Time Camera Planning

3.1 Problem Description

Controlling a camera from a third person view is a challenging task because it is too complex for the user to control the avatar and the camera at the same time. The easiest way to position the camera is to fix the camera behind the avatar in the avatar's local coordinate system. However, there is no guarantee that the generated view will not be occluded by obstacles. The kind of occlusions usually creates undesirable interruptions on game playing and should be avoided whenever possible. Some games try to detect the occluded obstacle and change it to semi-transparent while others attempt to increase the inclined angle of the camera but without guarantee of success. Another common way is by positioning several cameras at known locations in a virtual environment. When an avatar enters a specific region, the camera is switched to a most appropriate predefined location. Although this method is computational efficient, it has the drawbacks of limited views and being scene-dependent.

We think that a third-person camera should follow the avatar and adjust its relative position and angle with respect to the avatar to achieve a good and coherent view. The camera should be controlled by the computer to satisfy hard constraints as well as soft constraints. The hard constraints include that the camera should not collide with obstacles and the view to the avatar should not be occluded by the objects in the environment. The soft constraints include that the view angle and view distance should satisfy the preference of a user as much as possible. Although an intercut is allowed to avoid occlusion when necessary, it should follow the idioms in cinematography [12] whenever possible. In addition, the available planning time for each frame should be limited by some desired frame rate (such as at least 15 frames per second) for interactive applications. Therefore, it is crucial to design an efficient planning algorithm to generate the motion of the camera within a given time budget in order to interleave motion planning and graphics rendering without causing jerky control.

3.2 Definition of Configuration Space

In this paper, we use motion planning technique to generate camera motion in real time.. When treated as a free-flying object, a camera have six degrees of freedom: position (x, y, z), and orientation (roll, pitch, yaw). However, due to the curse of

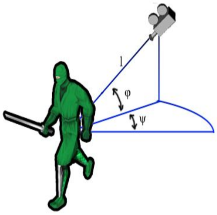


Fig. 1. Representation of camera configuration (*l*, ψ,φ)

dimensionality for the motion planning problem, searching the underlying six-dimensional space directly could be too complex to do in real time. Nevertheless, a regular camera rarely uses the roll DOF because of the disorder that it may cause. In [14], the authors only use three positional DOF to define a camera configuration because they assume that the camera always pointed to the avatar and position it at the center of the view. Similarly, we consider the position of the camera only but we used the relative coordinate frame with respect to the avatar in a polar coordinate system defined by distance *l*, horizontal angle ψ, and vertical angle φ, as shown in Fig. 1. Although both representations are equivalent, the relative coordinate system provides a more intuitive way for defining constraints or preferences.

3.3 Construction of Probabilistic Roadmap

The planning algorithm that we have used in our system is the Probabilistic Roadmap Method (PRM), one of the most popular methods in recent years for motion planning problems in high-dimensional configuration space. It has the advantage of being probabilistic complete and can be constructed incrementally, which is what we need in our real-time application. In a traditional PRM planner, an initial number of nodes are randomly generated in the free space of the configuration space, and nearby nodes are further connected by local paths, typically straight-line segments. For a given initial and a goal configuration, we first add them into the roadmap and then search for a continuous path in the roadmap connecting the two configurations. If one exists, the path is found; otherwise, more nodes can be added into the roadmap to enhance its connectivity.

In our real-time camera planning problem, we have used a roadmap update strategy similar to the one called Lazy PRM, proposed in [6][7]. The flowchart of the algorithm is shown in Fig. 2. A main feature of this approach is that it does not check collisions for a static roadmap in a preprocessing step like traditional PRM planners. Instead, the validity (legality) of a node or a link is checked only when the node or the link is under consideration during the search. Therefore, only the nodes that are related to finding a specific path at the moment need to be checked. Another difference of our planner is that the roadmap is described relative to the avatar and updated as the avatar moves, as shown in Fig. 3. Therefore, the roadmap is dynamic in nature even though there are no dynamic obstacles in the virtual scene. Nevertheless, since the motion of the avatar is continuous, the change of the roadmap is also incremental, which makes the real-time update of the roadmap feasible.

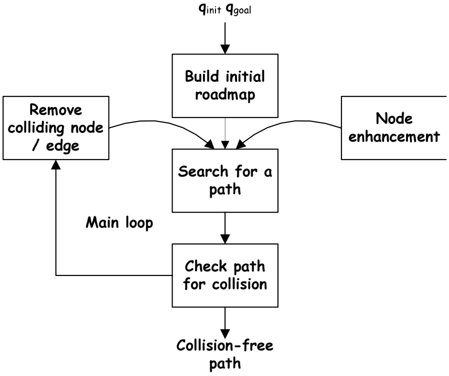


Fig. 2. Flowchart of the Lazy PRM algorithm

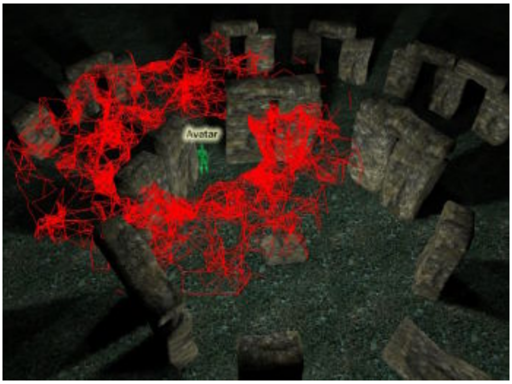


Fig. 3. An example of probabilistic roadmap computed with respect to the avatar

3.4 Camera Planning Algorithm

In a typical path planning problem, the inputs are an initial and a goal configuration in addition to the geometric description of the obstacles, and the output is a continuous path in the free space connecting the initial and the goal configurations. However, the camera planning problem differs from the classical motion planning problem in a few aspects. First, the goal is not clearly defined. In addition, there is no guarantee that a feasible path can be found in the given time budget. We will first describe the planning algorithm first and then describe the criteria that have been used to search for a feasible configuration.

In Fig. 4, we list the pseudo code for the time-budgeted camera planning algorithm. In this algorithm, we assume that a probabilistic roadmap with an appropriate number of nodes has been built in the configuration space. Instead of searching for the goal, the planner searches for a configuration that has the lowest cost for some criteria within a given time budget (TIME_LIMIT). The search starts from the current configuration and explore its neighbors in a best-first fashion. New legal neighbors will

```
Algorithm: Time_Budget_Based_Path_Search
1  $OpenList \leftarrow \{N_{init}\}$
2  $Timer \leftarrow 0$; $VisitedNode \leftarrow$ null
3  while $OpenList \neq \{\}$ and $Timer < TIME\_LIMIT$
4     $N_c$ = extractMin($OpenList$)
5     if isVisited($N_c$)
6     then continue
7     else $VisitedNode \leftarrow VisitedNode \cup N_c$
8     foreach neighbor $N_s$ of $N_c$
9      if isConnected($N_s$) = false
10        then continue
11     new_cost $\leftarrow$ Cost_Function($N_s$)
12     if cost[$N_s$] not nil and new_cost < cost[$N_s$]
13     then cost[$N_s$] $\leftarrow$ new_cost
14        $OpenList \leftarrow OpenList \cup N_s$
15        if cost[$N_s$] < cost($CUR\_BEST\_NODE$)
16        then $CUR\_BEST\_NODE \leftarrow N_s$
17 return $CUR\_BEST\_NODE$
```

Fig. 4. Time budget based camera planning algorithm

then be kept in the priority queue called OpenList, where nodes are sorted by the cost function (line 11) to be described in the next subsection. When the time is allowed, the node with the lowest cost in OpenList is chosen for further expansion. When the time is up or the OpenList becomes empty, the configuration CUR_BEST_NODE is returned and then the next configuration for the camera to take can be determined by back-tracking the path to the initial configuration.

Since the avatar is not static, the validity of a node or a link in a roadmap is also dynamic at run time. We use a linear dead-reckoning algorithm to predict the motion of the avatar in the next few steps. We augment the roadmap by an additional time dimension such that, at any given time, we can check the validity of a node according to this prediction but in a lazy manner, which means that we will validate a node or a link only when it is necessary. Although linear extrapolation may not be the best way to predict the motion of an avatar when the time is far in the future, the potential errors may not be critical since they can be recovered quickly when the avatar's motion is updated in the future.

3.5 Design of Cost Functions

The quality of a camera view is a subjective matter although there exist cinematographic principles that can be used for evaluation. In this work, we have defined a cost function composed of three components to allow the user to specify their preferences. The first component, f_1, is composed of more subjective parameters that the user prefers when viewing the avatar, as shown below.

$$f_1(q) = w_h \cdot d_h(q) + w_a \cdot d_a(q) + w_l \cdot d_l(q), \tag{1}$$

where the three difference functions are:

- **Height difference** (d_h)**:** The difference between the current height and the ideal height of the camera.
- **View angle difference** (d_a)**:** The angle of the current camera from the line of interest (LOI), which is the direction right behind the avatar in the tracking mode.
- **Distance difference** (d_l)**:** The difference between the current distance and the ideal distance from the avatar.

For a given configuration q, f_1 is computed according to these three functions and their corresponding weights specified by the user.

The second component f_2 is computed based on visibility as shown below.

$$f_2 = \sum_{i=1}^{n} \alpha^i O_{occ}(q, t+i)\,, \tag{2}$$

where $C_{occ}(q, t)$ is a constant occlusion cost for q at a give time t, and α is a decaying factor ($0 < \alpha < 1$), and n is the number of time steps under consideration for q. In other words, a configuration will be checked for occlusion for a period of time with decreasing weights in the future.

The third component f_3 is the distance cost of camera configurations in the roadmap:

$$f_3(q) = \text{distance}(q, q_s), \tag{3}$$

where q_s is the start camera configuration in the current search. This component allows the user to specify his/her preference of remaining unchanged in the current configuration.

The overall cost function $f(q)$ for a given camera configuration q is computed as a linear combination of these three components, where the weights (w_1, w_2, w_3) can also be chosen by the user to specify their emphasis on these criteria.

$$f(q) = w_1{\cdot}f_1(q) + w_2{\cdot}f_2(q) + w_3{\cdot}f_3(q) \tag{4}$$

4 Consideration of Camera Intercuts

Due to the time constraints and unpredictability of avatar motions, the real-time camera planner presented in the previous section does not guarantee to find an non-occluded view and may take the camera to a difficult situation (such as the example shown in Fig. 5) where a legal camera configuration with continuous movement is not possible in the future. In this case, an intercut that allows the camera to jump to

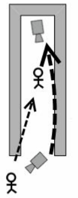


Fig. 5. Illustration of entering a deadend when intercut is needed

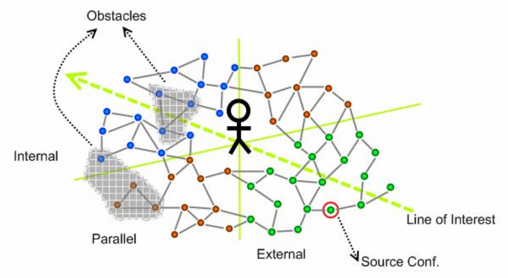


Fig. 6. Classification of nodes in the roadmap according to LOI

another appropriate location might be more desirable. In this section, we will describe how we account for this type of camera operation in the current planning algorithm.

In order to account for intercuts, we need to consider more than one camera. We can plan two cameras with two different sets of cost functions at the same time and then switch between them when the current camera is trapped in a difficult situation. However, the planning time will double in this case. Instead, we propose to use the concept of virtual links in the roadmap to consider both cameras in the same data structure. A virtual link is a temporary link from the start configuration to another disconnected configuration. The virtual links are constructed every time before the search starts. A virtual link is considered legal only if it satisfies the following conditions:

- The other end of a virtual link must be a legal node (non-occlusive and collision-free).
- Both ends of a virtual link must lie in the same side of the Line of Interest, which is defined as the heading direction of the avatar in this case.
- The other end of a virtual link must lie in a region that is far enough from the start configuration; otherwise, the user may experience more a jump cut instead of an intercut.

In order to facilitate the selection of the other end of a virtual link, we divide the roadmap into six different regions with three regions at one side of LOI as shown in Fig. 6. These regions (internal, parallel, and external) are defined according to the view angles relative to the heading orientation of the avatar. A valid virtual link must have their end points from two different regions, and the distance between the two ends must be above some threshold.

After a given number of virtual links are computed for the start configuration, the search can proceed as before except for that the cost of performing an intercut is made higher than regular movements. We modify the cost function f_3 to take this case into account. The modified f_3 is as follow.

$$\begin{array}{lll} f_3(q) & = C_{cut}, & \text{if a virtual edge} \\ & = \text{distance}(q, q'), & \text{otherwise} \end{array} \qquad (6)$$

The movement cost C_{cut} for an intercut should be higher than the regular movement cost, and its value should be a time-dependent variable. In our design, we have used the following formula to define C_{cut}.

$$\begin{aligned} C_{cut} &= C_{cut}^{max}, && \text{right after the intercut} \\ &= \max(C_{cut}^{min}, C_{cut} - 1) && \text{otherwise,} \end{aligned} \qquad (7)$$

where C_{cut}^{max} is the cost right after the intercut is performed. The cost will decrease gradually as the time passes by.

5 Experimental Results

We have developed the camera planning module in C++ in an experimental system based on the Object-Oriented Graphics Rendering Engine (OGRE) 3D graphics engine [1]. A collision detection module with simple hierarchical bounding boxes has also been implemented to detect the possible collisions of the avatar and the camera with the obstacles in the environment. In addition, to simplify computation, we have used four rays shooting from the camera to the avatar to detect occlusions. All experiments were conducted on a regular PC with a Pentium 4 3.0GHz CPU and 1GB of RAM. The allocated maximal time budget for planning computation in each frame is 65ms but actually the frame rate is maintained at around 60fps in all experiments. A constant number of 900 nodes are maintained for the dynamic roadmap occupying a region with a radius of around one fourth of the workspace of the virtual scene.

5.1 Basic Experiments

We have used a test scene in Fig. 7 to see if the camera can move to avoid occlusions with the obstacles in the environment. In this experiment, the main focus is on occlusion, and thus no special preference has been specified. As shown in Fig. 7, the scene consists of a circular array of huge stones that are likely to block the camera view when the avatar moves around the stones. In order to repeat the experiments for different parameters, we record the path of the avatar ahead of time (as shown in the right of Fig. 7) and replay the motion as a simulation in the experiments for different camera parameters. In Fig. 8(b), we show the snapshots of the views acquired from the simulation experiment at various locations defined in Fig. 8(a). In general, the camera can successfully track the avatar except for some short period of time when the avatar turned very sharply. In this case, the view was slightly occluded.

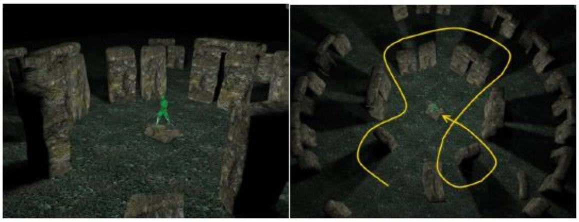


Fig. 7. Inclined view of the test scene and a sample path taken by the avatar

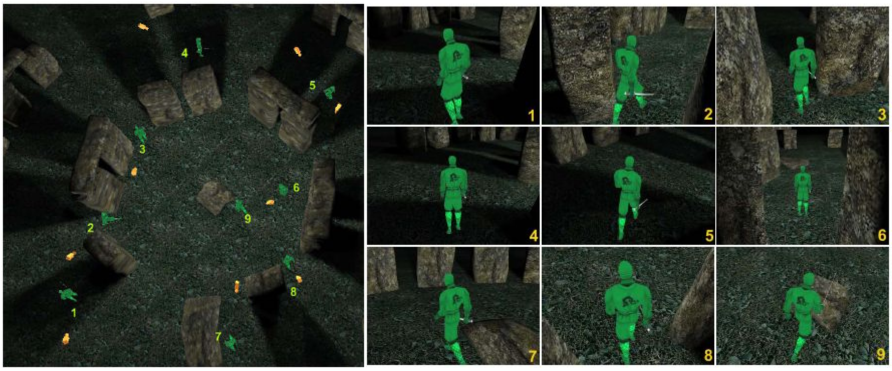


Fig. 8. Snapshots in part (b) are taken from locations at part (a)

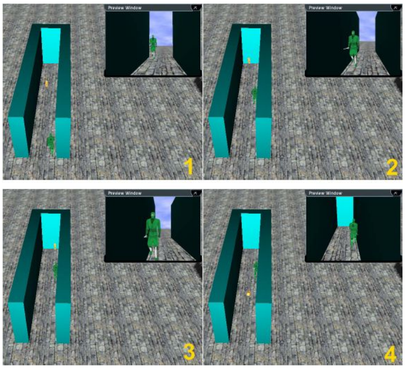


Fig. 9. Snapshots of the camera undergoing an intercut

We also have designed the scene in Fig. 5 to intentionally trap the camera inside the dead end. In this case, the only way for the camera to avoid being occluded or getting into the obstacles is by an intercut. The experimental result is shown in Fig. 9. In the first snapshot, the camera entered the corridor first with an external view. As the avatar moved closer to the dead end of the corridor in the second and the third snapshots, the camera had fewer and fewer legal configurations that the camera can move to without jumps. Therefore, an intercut view was selected eventually by the planner, as shown in the fourth snapshot, to avoid bumping into the walls.

5.2 Experiments on Prediction Depths

The depth of the prediction for avatar motion that can be reached within a given time budget affects the final result. We have done an experiment to study this factor. The test scene is a maze-like environment where the camera needs to track the avatar closely to avoid occlusions shown in Fig. 10. The prediction of the avatar's future motion is also crucial for avoiding occlusions. We have used five different sets of preference parameters (exp1-exp5) to run the experiments with four different values

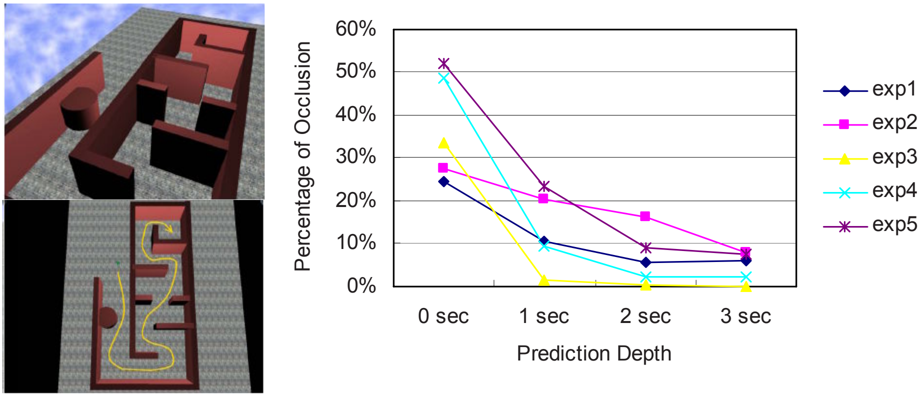


Fig. 10. Test scene and the experimental results of how prediction depth affects the percentage of occlusion time during several runs of navigation with different preference settings

of prediction depths (0, 1, 2, 3 seconds). These five different sets of parameters were chosen based on the following combinations: 1) no special preferences; 2) a middle shot from a fixed distance; 3) a close shot from an external view; 4) a middle shot from an external view; and 5) a long shot. At the right of Fig. 10, we show the plot of percentage of time that the view is occluded during the navigation for four prediction depths with five different preference settings. We found that prediction of the avatar's motion indeed decreased the percentage of occlusion time for all preference settings but the effect degrades quickly when the prediction depth is longer than one second. This result suggests that using the concept of time budget and motion prediction to plan camera motions is an effective way to automate the generation of camera motion, and the prediction depth does not need to be long.

6 Conclusions

It is highly desired in real-time applications such as 3D games to have the camera motions generated automatically in a third-person control mode. In this work, we have successfully used a lazy PRM method developed originally for robotics to maintain a dynamic probabilistic roadmap specified relative to the avatar. As the avatar moves, the roadmap is maintained in a lazy fashion so that the validity of the nodes and links are checked only when they are visited. We also have proposed to use virtual links to account for intercuts in the search for a legal and high-quality move. The user is allowed to specify their viewing preferences or styles through the parameters that we have designed. We have tested the planner in a real-time 3D virtual environment with different scenes, and satisfactory results have been obtained and reported.

In the future, we are planning to investigate the possibility of using more sophisticated motion prediction methods for the avatar to understand its effect on the resulting camera motions. We would also like to study how to extend the work to consider more sophisticated viewing objectives in addition to tracking the avatar during navigation. These viewing objectives may be implicitly or explicitly specified in the environment under the concept of semantic virtual environment.

Acknowledgments

This research was funded in part by the National Science Council (NSC) of Taiwan under contract no. NSC 95-2221-E-004-015.

References

1. Arijon, D.: Grammar of the Film Language. Hastings House Publishers (1976)
2. Bares, W., Gregoire, H.J.P., Lester, J.C.: Realtime Constraint-Based Cinematography for Complex Interactive 3D Worlds. In: Proc. of the Tenth Conf. on Innovative Applications of Artificial Intelligence (1998)
3. Bares, W., McDermott, S., Boudreaux, C., Thainimit, S.: Virtual 3D camera composition from frame constraints. In: Proc. of the ACM Intl. Conf. on Multimedia, pp. 177–186. ACM Press, New York (2000)
4. Bares, W., Thainimit, S., McDermott, S.: A Model for Constraint-Based Camera Planning. In: Proc. of the 2000 AAAI Spring Symposium (2000)
5. Blinn, J.F.: Jim blinn's corner: Where am I? what am I looking at? IEEE Computer Graphics and Applications 8(4), 76–81 (1988)
6. Bohlin, R., Kavraki, L.E.: Path planning using lazy PRM. In: IEEE Int. Conf. on Robotics & Automation, pp. 521–528 (2000)
7. Bohlin, R., Kavraki, L.E.: A Lazy Probabilistic Roadmap Planner for Single Query Path Planning. In: Proc. of IEEE Int. Conf. on Robotics and Automation (2000)
8. Bourne, O., Sattar, A.: Applying Constraint Satisfaction Techniques to 3D Camera Control. In: 17th Australian Joint Conf. on Artificial Intelligence (2004)
9. Christianson, D.B., Anderson, S.E., He, L.W., Salesin, D.H., Weld, D.S., Cohen, M.F.: Declarative Camera Control for Automatic Cinematography. In: Proc. of the Thirteenth National Conf. on Artificial Intelligence (AAAI 1996) (1996)
10. Christie, M., Machap, R., Normand, J.M., Olivier, P., Pickering, J.: Virtual Camera Planning: A Survey. In: Butz, A., Fisher, B., Krüger, A., Olivier, P. (eds.) SG 2005. LNCS, vol. 3638. Springer, Heidelberg (2005)
11. Halper, N., Helbing, R., Strothotte, T.: A Camera Engine for Computer Games: Managing the Trade-Off between Constraint Satisfaction and Frame Coherence. In: Proceedings of Eurographics 2001 On Computer Graphics Forum, vol. 20(3), pp. 174–183 (2001)
12. He, L.W., Cohen, M.F., Salesin, D.H.: The Virtual Cinematographer: A Paradigm for Automatic Real-Time Camera Control and Directing. In: Proc. of the 23rd Annual Conf. on Computer Graphics and Interactive Techniques (1996)
13. Latombe, J.: Robot Motion Planning. Klumer, Boston (1991)
14. Li, T.Y., Yu, T.H.: Planning Tracking Motions for an Intelligent Virtual Camera. In: Proc. of IEEE Int. Conf. on Robotics and Automation (1999)
15. Lin, T.C., Shih, Z.C., Tsai, Y.T.: Cinematic Camera Control in 3D Computer Games. In: The 12th Int. Conf. Central Europe on Computer Graphics, Visualization and Computer Vision (2004)
16. Nieuwenhuisen, D., Overmars, M.H.: Motion Planning for Camera Movements in Virtual Environment. In: Proc. IEEE Int. Conf. on Robotics and Automation (2004)
17. Oliveros, D.A.M.: Intelligent Cinematic Camera for 3D Games. Thesis, Univ. of Technology, Sydney Australia (2004)
18. Salomon, B., Garber, M., Lin, M.C., Manocha, D.: Interactive Navigation in Complex Environment Using Path Planning. In: Proc. of the 2003 symposium on Interactive 3D graphics (2003)
19. OGRE 3D, `http://www.ogre3d.org/`

Virtual Camera Composition with Particle Swarm Optimization

Paolo Burelli[1], Luca Di Gaspero[2], Andrea Ermetici[1], and Roberto Ranon[1]

[1] HCI Lab, University of Udine, via delle Scienze 206, 33100, Udine, Italy
roberto.ranon@dimi.uniud.it
[2] DIEGM, University of Udine, via delle Scienze 208, 33100, Udine, Italy
l.digaspero@uniud.it

Abstract. The Virtual Camera Composition (VCC) problem consists in automatically positioning a camera in a virtual world, such that the resulting image satisfies a set of visual cinematographic properties [3]. In this paper, we propose an approach to VCC based on *Particle Swarm Optimization* [5]. We show, in realistic situations, that our approach outperforms a discretized, exhaustive search method similar to a proposal by Bares et al [1].

1 Introduction

In 3D graphics interactive applications, effective camera placement and control is fundamental for the user to understand the virtual environment and be able to effectively accomplish the intended task. For example, in 3D information visualization, bad camera placements could cause the user to miss important visual details (e.g., because they are occluded) and thus make wrong assumptions on the data under analysis.

In most current 3D applications, users directly position the camera using a input device through a tedious and time-consuming process requiring a succession of "place the camera" and "check the result" operations [3]. In recent years, some researchers (e.g., [1, 3, 6, 8]) have come up with methods to automatically position the camera that aim at relieving the user from direct control, and are inspired by how human cinematographers approach the same problem: first, requirements on the image or shot to be obtained are stated (e.g., by the user), and then, a camera fulfilling those requirements is computed.

More specifically, the Virtual Camera Composition (VCC) problem consists in positioning a camera in a virtual world, such that the resulting image satisfies a set of visual cinematographic properties [3], e.g. subjects' size and location. The approaches developed so far typically model VCC as a constraint satisfaction or optimization problem (some approaches use both) where the desired image properties are represented as constraints or objective functions. A range of different solving techniques [4] have been explored in the past, but generating effective results in real-time (or near-real time), even with static scenes, remains an issue. Since this requirement is very important in interactive applications, there is the need of finding more efficient methods, as well as to compare the performances of previously proposed approaches in realistic contexts.

In this paper, we present and evaluate an approach to VCC that employs *Particle Swarm Optimization* (hereinafter, PSO) [5], a method which, to the best of our

A. Butz et al. (Eds.): SG 2008, LNCS 5166, pp. 130–141, 2008.

knowledge, has never been applied to this kind of problems. PSO is a population-based method for global optimization, which is inspired by the social behavior that underlies the movements of a swarm of insects or a flock of birds. We will show how the PSO approach can be used to solve VCC problems in the case of static scenes, and evaluate its performances against a discretized, exhaustive search approach very similar to the one proposed by Bares et al in [1]. Finally, one of our motivations for this research is to use automatic cameras to support users' navigation in virtual environments. Although this is not the focus of this paper, we will briefly present the idea in our experimentation.

The paper is organized as follows. Section 2 reviews related work, while Section 3 describes our approach to VCC. Section 4 presents the experimental results. Finally, in Section 5 we conclude the paper and outline future work.

2 Related Work

A comprehensive survey of approaches to camera control can be found in [4]. In the following, we focus on approaches to VCC that: (i) employ a declarative "cinematographic" style, i.e. where the problem is expressed as a set of requirements on the image computed from the camera, such as relative viewing angles and occlusions, and (ii) model the problem as a constraint and/or optimization system. These approaches are by far the most general, and therefore interesting for a wide range of applications.

In constrained search and/or optimization approaches to VCC, the properties of the image computed from the camera are expressed as numerical constraints on the camera parameters, (typically, camera position, orientation, and FOV). Although pure optimization (e.g., [6]) or constraint satisfaction (e.g., [2]) have been used in the past, more recent proposals tend to adopt an hybrid strategy, where some requirements are modeled as constraints, and used in a first phase to compute geometric volumes of feasible camera positions [1, 3, 8]. Then, in a second phase, requirements are modeled as objective functions that are maximized by searching inside the geometric volumes using various optimization methods, such as stochastic [3, 8] or heuristic [1] search.

The advantage of the hybrid approach is that it can reduce complexity by limiting the search space using geometric operators to implement constraints in the first phase, and, at the same time, the optimization phase can increase the chances (with respect to a pure constraint-based approach) that a (possibly good) solution will be found: in many situations, it is better to have a solution, although not satisfying some requirements, than having no solution at all. Another advantage is the fact that the geometric volumes generated in the first phase can be semantically characterized with respect to their visual properties [3], e.g. to allow the computation of multiple, potentially equivalent solutions instead of just generating a single one.

3 A PSO Approach to VCC

In this Section, we describe how PSO can be used to solve the VCC problem. First, we will present the language that can be used to describe the properties of the image to be generated from the camera. Our language comprises most of the visual properties from prior work including [1, 3], so, for reasons of space, we refer the reader to those papers for a more detailed explanation. Then, we will describe the solving process. As other approaches mentioned in the previous section, we follow a hybrid strategy. Therefore, we first compute volumes of feasible camera positions from some of the requirements that constrain the position of the camera. This phase of the solving process is quite similar to [3] (our proposal can thus be considered a variation of that approach), so we will just give an high-level overview. The search inside the feasible regions is then carried out with PSO, on which we will focus in detail. As in all other approaches to VCC, we consider a classical pinhole Euler-based camera model where the parameters are the camera position, orientation, and FOV. Finally, the visual properties we adopt (as well as the solving method) are not meant for dynamic scenes with temporary occlusions, which requires properties expressed over more than just the current point in time.

3.1 Available Image Properties

The following is the list of properties that involve an object in the scene. Most properties include a real argument, w, whose value encodes the importance of the requirement.

- **Object view angle.** Requires the camera to lie at a specified orientation relative to an object: $objHAngle(\textsf{Object}\, x, \textsf{angle}\, \theta, \textsf{angle}\, \gamma, \textsf{double}\, w)$, where θ is the angle between the object front vector and the preferred viewing direction, and γ defines a range of allowed angles around the preferred direction; $objVAngle(\textsf{Object}\, x,$ $\textsf{angle}\, \theta, \textsf{angle}\, \gamma, \textsf{double}\, w)$, where θ is the angle between the object up vector and the preferred viewing direction, while γ defines a range of allowed angles around the preferred direction;
- **Object inclusion in the image.** Requires a specified fraction $f \in [0, 1]$ of the object to lay inside the FOV of the camera: $objInFOV(\textsf{Object}\, x, \textsf{double}\, f, \textsf{double}\, w)$ ($f = 0$ means the object must not be in the camera FOV);
- **Object projection Size.** Requires the projection of the object to cover a specified fraction $f \in]0, 1]$ of the image: $objProjSize(\textsf{Object}\, x, \textsf{double}\, f, \textsf{double}\, w)$;
- **Object distance from camera.** Requires the object to lie at a specified distance from the camera: $objDistanceFromCam(\textsf{Object}\, x, \textsf{double}\, d_{min}, \textsf{double}\, d_{max})$;
- **Object Position in Frame.** Requires a specified fraction $f \in [0, 1]$ of the object to lie inside a given rectangular subregion of the image: $objProjPosition(\textsf{Object}\, x,$ $\textsf{2DPoint}\, p_1, \textsf{2DPoint}\, p_2, \textsf{double}\, f, \textsf{double}\, w)$, where p_1, p_2 are two points in the image identifying the top-left and bottom-right corners of the rectangular region (0 means the object must not be inside the rectangle);
- **Object Occlusion.** Requires the specified fraction $f \in [0, 1]$ of the object projection to be occluded in the image: $objOcclusion(\textsf{Object}\, x, \textsf{double}\, f, \textsf{double}\, w)$ (0 means the object should be not occluded at all).

Additionally, a set of camera-related properties are defined. They are useful to directly limit the search space when suitable; for example, upside-down cameras and cameras placed inside walls or other objects are typically unsuitable.

- **Camera outside region.** Requires the camera to lie outside a box-shaped region in 3D space: $camOutsideRegion(\textsf{3DPoint}\, p_1, \textsf{3DPoint}\, p_2)$, where p_1, p_2 are two opposite corners of the box.

- **Camera outside object.** Requires the camera to lie outside the bounding box of a specified object: $camOutsideObj(\textsf{Object}\,x)$.
- **Camera above plane.** Requires the camera to lie above a given plane in 3D space: $camAbovePlane(\textsf{3DPoint}\,o, \textsf{3DPoint}\,n)$, where o is a point on the plane and n is the normal of the plane.
- **Bind camera parameters:** $camBindX(\textsf{double}\,x_{min}, \textsf{double}\,x_{max})$, $camBindY(\textsf{double}\,y_{min}, \textsf{double}\,y_{max})$, $camBindZ(\textsf{double}\,z_{min}, \textsf{double}\,z_{max})$ $camBindRoll(\textsf{angle}\,\alpha_{min}, \textsf{angle}\,\alpha_{max},)$, $camBindYaw(\textsf{angle}\,\alpha_{min}, \textsf{angle}\,\alpha_{max},)$, $camBindPitch(\textsf{angle}\,\alpha_{min}, \textsf{angle}\,\alpha_{max},)$, $camBindFOV(\textsf{angle}\,\alpha_{min}, \textsf{angle}\,\alpha_{max},)$, $camAspectRatio(\textsf{double}\,f)$.

All these atomic properties can be combined to form more complex descriptions by using the logic operator $\wedge$.

3.2 Phase 1: Computing Volumes of Feasible Camera Positions

In this phase we use some of the specified properties as geometric operators to derive (possibly non connected) volumes in 3D space where it is feasible to position the camera. More particularly, the properties which are considered in this phase are:

- Object-related properties: **Object View Angle**, **Distance from Camera**;
- Camera-related properties: **Camera outside region**, **Camera outside object**, **Camera above plane** and **Bind camera parameters**.

For example, a **Distance from camera** property defines a feasible volume which is the difference of two spheres, whose center is the center of the bounding volume of the considered object, and whose radii are respectively the maximum and minimum allowable distances. The geometric operations for the other properties are defined in detail in [1, 3]. Note that, contrary to [3], we do not employ occlusion-related properties in this step because we prefer to avoid cutting too much the search space (with the risk of not generating any solution). The feasible volumes defined by each property are then combined with intersection operators to derive the (possibly non-connected) feasible volume into which the search with PSO will be carried out.

Technically, this phase has been implemented using the *VTK* library(`www.vtk.org`) that provides the required geometric primitives (planes, boxes, ...) and operators (intersection, difference, ...) to derive the feasible volumes of space as implicit functions. The optimization phase will then use the computed implicit functions to evaluate when a point lies inside or outside the feasible volume.

3.3 Phase 2: Searching Inside the Feasible Volume with PSO

Swarm Intelligence is an Artificial Intelligence paradigm, which relies on the exploitation of the (simulated) behavior of self-organizing agents for tackling complex control and optimization problems. In particular, PSO [5] is a population-based method for global optimization, whose dynamics is inspired by the social behavior that underlies the movements of a swarm of insects or a flock of birds. These movements are directed toward both the best solution to the optimization problem found by each individual and the global best solution.

More formally, given a D-dimensional (compact) search space $S \in \mathbb{R}^D$ and a scalar objective function $f : S \to \mathbb{R}$ that assesses the quality of each point $x \in S$ and has to be maximized, a *swarm* is made up of a set of N particles, which are located in that space. The i-th particle is described by three D-dimensional vectors, namely:

- The particle current *position* $\mathbf{x}_i = (x_{i_1}, x_{i_2}, \ldots, x_{i_D})$;
- The particle *velocity* $\mathbf{v}_i = (v_{i_1}, v_{i_2}, \ldots, v_{i_D})$, i.e., the way the particle moves in the search space;
- The particle *best visited position* (as measured by the objective function f) $\mathbf{P}_i = (p_{i_1}, p_{i_2}, \ldots, p_{i_D})$, a memory of the best positions ever visited during the search.

The index of the particle that reached the global best visited position is denoted by g, that is, $g = \arg\max_{i=1,\ldots,N} f(\mathbf{P}_i)$.

At the beginning of the search (step $n = 0$), the particles are set at random locations and with random velocities. The search is performed as an iterative process, which at step n modifies the velocity and position vectors of each particle on the basis of the values at step $n - 1$. The process evolves according to the following rules (superscripts denote the iteration number):

$$\mathbf{v}_i^n = w^{n-1}\mathbf{v}_i^{n-1} + c_1 r_1 \left(\mathbf{p}_i^{n-1} - \mathbf{x}_i^{n-1}\right) + c_2 r_2 \left(\mathbf{p}_g^{n-1} - \mathbf{x}_i^{n-1}\right) \tag{1}$$

$$\mathbf{x}_i^n = \mathbf{x}_i^{n-1} + \mathbf{v}_i^n \qquad i = 1, 2, \ldots, N \tag{2}$$

In these equations, the values r_1 and r_2 are two uniformly distributed random numbers in the $[0, 1]$ range, whose purpose is to maintain population diversity. Constants c_1 and c_2 are respectively the so-called *cognitive* and *social* parameters, which are related to the speed of convergence. The value w^n is an *inertia weight* and it establishes the influence of the search history on the current move. A high weight is related to a global exploration, while a low weight allows a local exploration (also called exploitation). In Section 4, we will discuss the values we have chosen for all the parameters (including the number of particles and iterations).

Since at each iteration a solution to the problem is available (although it could not be the optimal one), PSO belongs to the family of *anytime* algorithms, which can be interrupted at any moment still providing a solution. In the general case, however, the iterative process is run until either a pre-specified maximum number of iterations has elapsed or the method has converged (i.e., all velocities are almost zero).

Having overviewed how PSO works, we now describe how we use it for searching inside the feasible volumes computed in the previous phase. In our case, the PSO search space is composed by all camera parameters, so it is a subset of $\mathbb{R}^7$. In particular, each particle position in $\mathbb{R}^7$ completely defines a camera, i.e. it assigns a value to each of the 7 camera parameters (3 position coordinates, 3 orientation angles, and FOV angle):

- The first three dimensions (x_0, x_1, x_2) correspond to the camera position. These values will be kept inside the feasible volume during the optimization process;
- The second three dimensions (x_3, x_4, x_5) correspond to the camera orientation, each ranging from 0 to 2π;

- The last dimension (x_7) corresponds to the camera FOV, ranging from a minimum to a maximum value;

In other words, PSO will optimize all camera parameters, but, with respect to the camera position (first three dimensions of the search space), search will be restricted inside the feasible volume computed in the previous phase. In practice, to check if a particle is inside or outside the feasible volume, we simply use x_0, x_1, and x_2 as arguments to the implicit functions derived in phase 1.

At the beginning of the search, N particles are set at random locations inside the feasible volumes (i.e., we randomly generate particles until we obtain N of them inside the feasible volume). Since particles might exit the feasible volume during the execution of PSO, after each iteration we check each particle with respect to the implicit functions (i.e., we check the position of the camera defined by the particle) and, if the particle is not in the feasible volume, we assign it the worst possible value of the objective function (i.e., 0). This will make the particle return into the feasible volume in the next iterations.

The objective function is built by taking into account the following object-related properties: **Object view angle**, **Object inclusion in the image**, **Object projection Size**, **Object Position in Frame**, and **Object Occlusion**. The degree of satisfaction of each property is evaluated by means of a function $f : \Pi \times S \rightarrow [0, 1]$ whose semantics depends on the property $\pi \in \Pi$ at hand. In general, the function measures a relative difference between the desired value for the property and its actual value. The value of f is then normalized in order to obtain a real value in the range $[0, 1]$, where 1 represents the satisfaction of the associated constraint and 0 is complete unfulfillment. For example, to evaluate an Object Position in Frame Property, we calculate the projection of the object bounding sphere, and then determine how much this projection covers the specified rectangular region. In the case of an occlusion property, we calculate the value of f by ray casting from the camera to the bounding box of the object (one to the center of the bounding box, and 8 to its corners), and testing intersections of the rays with other objects in the scene (where the number of rays intersecting other objects gives the percentage of occlusion). This is not optimal, since it might be problematic for large occluders and not very precise with respect to the expression of partial occlusion. However, how the satisfaction of each property is calculated is independent from PSO, and can thus be improved maintaining the general solving process. For the other properties, the calculation of the objective function is similar to the ones described in [1, 3, 4].

When atomic properties are combined by $\wedge$ connectors, the combined objective function is computed as $f(\pi_1 \wedge \pi_2, \mathbf{x}) = w_1 f(\pi_1, \mathbf{x}) + w_2 f(\pi_2, \mathbf{x})$, where the weights w_i are those given as last argument to each property.

In general, the evaluation of a particle requires the function f to be computed for all the properties of the image description. However, this process can be quite time consuming, especially in the case of complex image descriptions or with properties that require a computationally intensive evaluation (e.g., occlusion). Therefore, we adopt a *lazy* evaluation mechanism for the objective function: the computation of f for the particle i is stopped if the sum of the weights of the properties that still have to be evaluated is smaller than the best objective value $f(\pi, \mathbf{P}_i)$. Therefore, as a heuristic, it could be useful to sort the properties leaving at the end the ones whose evaluation has a higher computational cost.

4 Implementation and Experimental Results

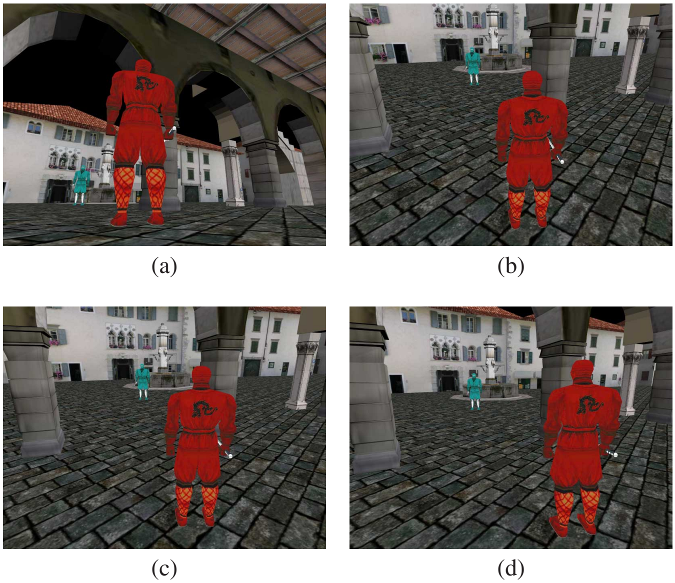
The VCC approach described in the previous section has been implemented as a part of a general C++ library which can be used in 3D applications. The library includes, besides the PSO solver, a discretized, exhaustive search-based algorithm (hereinafter called EXH) which follows the approach described in [1]. The first phase is identical for PSO and EXH. In the second phase, EXH scans a grid of $n \times m \times o$ points in the 3D scene (where n, m and o can be varied depending on the desired resolution). For each point in the grid, when it is inside the feasible volume, EXH considers a finite set of camera orientations (with differ by 15 degrees in each Euler angle) and FOV values, and for each one it evaluates the objective function exactly as PSO (including lazy evaluation). After having considered all points, or when the value of the objective function is sufficiently close to the optimum, EXH returns the best found camera.

In our experimentation, we will consider three increasingly complex VCC problems and show the resulting cameras generated by PSO and EXH. To make the test more realistic, we have set all the problems in the 3D model of a medieval village (which can be visited at `udine3d.uniud.it/venzone/en/index.html`), so that in evaluating occlusions the algorithms will have to take into account several objects.

All the VCC problems have been solved 100 times each with both algorithms. Since the PSO approach can produce different cameras in different runs of the same problem, we will show more cameras for each problem solved by PSO, and just one camera for EXH (which calculates the same camera in each run). Then, we will discuss the computational performances. For PSO, we have employed in each problem a population of 64 particles and limited the maximum number of iterations to 50. These parameter values have been determined through experimentation, and have demonstrated to be fairly robust in all tests we have made. For the other parameters of PSO, we have employed $c_1 = c_2 = 0.5$ and an inertia weight w which linearly decreases from an initial value of 1.2 to 0.2, in order to balance between exploration and exploitation at different stages of the search. These values are considered a good choice in many problems to which PSO has been applied.

For EXH, we have employed in each test the smallest grid (in terms of number of points) such that the algorithm was able to generate a camera with fitness value close the one PSO was able to compute. Moreover, both approaches have been set to stop the search when they reach the threshold of 98% of optimal value. Both algorithms have been compiled with Microsoft Visual Studio 2005 C++ compiler and run on an Athlon 64 X2 5000+ (2.4 GHz), 4 GB ram running Microsoft Windows XP.

In each of the following tests, we have introduced a set of properties to limit camera orientation and FOV to suitable values. More specifically, we lock the up vector of the camera parallel to the world Y axis and the FOV to the value currently being used by the rendering engine, and limit the camera position inside a box containing the whole scene. The common properties used in the tests presented in this paper are listed the following (in all listings we omit the $\wedge$ connectors).



(a) (b)

(c) (d)

Fig. 1. Cameras computed from the properties listed in 1.2: (a) camera computed by EXH considering a 25 × 25 × 25 grid of possible camera positions; (b,c,d) cameras computed by PSO in three different runs

Listing 1.1. Properties common to all problems

```
camBindRoll(0, 0)
camAspectRatio( currentScreenAspectRatio )
camBindFOV( currentCameraFOV )
camBindX( x_{min}, x_{max})
camBindY( y_{min}, y_{max})
camBindZ( z_{min}, z_{max})
```

Over the shoulder shot. The objective of this problem is to obtain a typical *over the shoulder* shot that shows the spatial relationship between two characters in the scene. The resulting cameras are shown in Figure 1.

Listing 1.2. Over the shoulder

```
camOutsideObject(blueWarrior)
objOcclusion(blueWarrior, 0.0, 1.0)
objInFOV(blueWarrior,1.0,1.0)
camOutsideObj(redWarrior)
objOcclusion(redWarior, 0.0, 1.0)
objInFOV(redWarrior,1.0,1.0)
objHAngle(redWarrior, -180, 90, 1.0)
objProjSize(redWarrior, 0.3, 1.0)
objVAngle(redWarrior, 45, 15, 1.0)
```

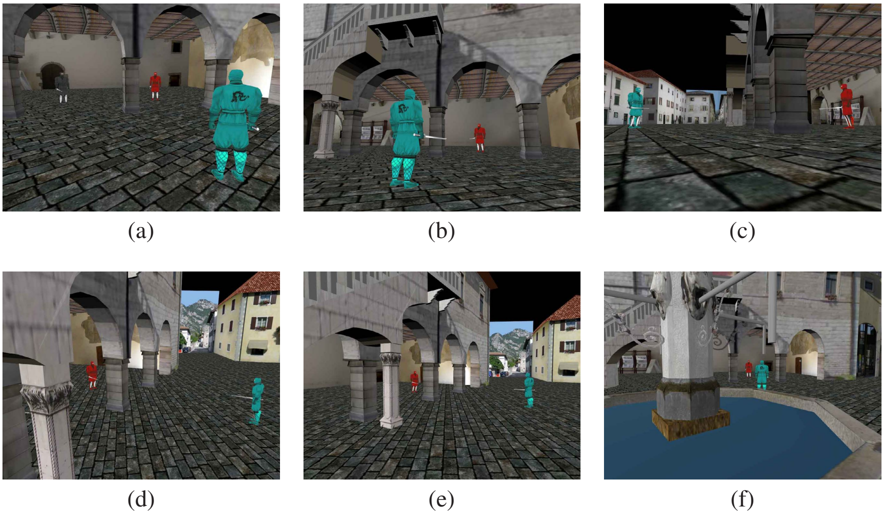

(a)
(b)
(c)
(d)
(e)
(f)

Fig. 2. Cameras computed from the properties listed in 1.3: (a) an image showing the relative position of three warriors in the scene; (b) camera computed by EXH considering a 10 × 10 × 10 grid of possible camera positions; (c,d,e,f) cameras computed by PSO in four different runs

Occlusion. The objective of this problem is to obtain an image where one of three characters in the scene (whose relative positions are shown in Figure 2a) is completely occluded, while the others are completely visible. The resulting images are shown in Figure 2. Note that, in this case, the cameras produced by PSO (Figures 2c and 2d) in different runs are quite different.

Listing 1.3. Occlusion

```
camOutsideObject(blueWarrior)
objOcclusion(blueWarrior, 0.0, 1.0)
objInFOV(blueWarior,1.0,1.0)
camOutsideObject(redWarrior)
objOcclusion(redWarior, 0.0, 1.0)
objInFOV(redWarrior,1.0,1.0)
objOcclusion(blackWarrior, 1.0, 1.0)
```

Navigation aid. One interesting possibility is to use automatically generated cameras to help users in orienting themselves and in reaching places of interest in virtual environments. For example, by augmenting the scene with semantic information about landmarks, points of interest, and paths, one could derive generic rules that compute (and present to the user during navigation) cameras that highlight the most important or closer navigational elements. This could help users to: (i) learn landmarks during exploration and therefore increase the acquisition of navigational knowledge; (ii) see and recognize landmarks during orientation and search. By also including the user's avatar in the computed image, it is also possible to highlight the current spatial relation between the user's actual position and the navigational elements in the scene. The

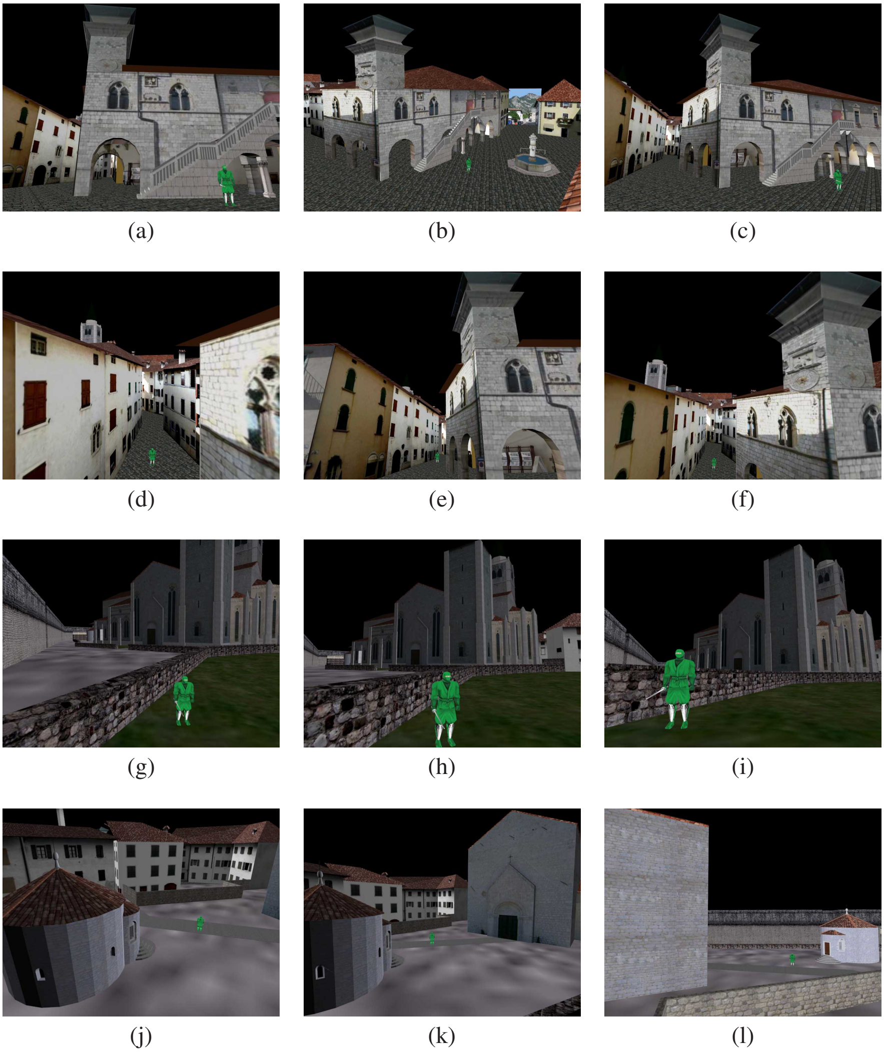


Fig. 3. Cameras computed from the properties listed in 1.4. Each row shows images computed with the user's avatar in different positions. For each row, the left image has been computed by EXH considering a $15 \times 15 \times 15$ grid of possible camera positions; the center and right images have been computed by PSO.

following problem regards a situation where we have some landmarks (the cathedral dome, its tower, the town hall tower and the baptistery) and we would like to compute cameras that shows the avatar at a certain size (most important requirement, otherwise the user could not be able to find its position), plus most possible landmarks. The results obtained with different position of the user's avatar (which is the green character in the images) are reported in figure 3. Note that requests cannot be fully satisfied in any run,

Listing 1.4. Navigation Aid

```
objProjSize(avatar,0.1,5.0)
objOcclusion(avatar, 0.0, 10.0)
objInFOV(avatar,1.0,12.5)
objOcclusion(baptistery, 0.0, 1.0)
objInFOV(baptistery,1.0,1.0)
objOcclusion(town_hall_tower, 0.0, 1.0)
objInFOV(town_hall_tower,1.0,1.0)
objOcclusion(cathedral_dome, 0.0, 1.0)
objInFOV(cathedral_dome,1.0,1.0)
objOcclusion(church_tower, 0.0, 1.0)
objInFOV(church_tower,1.0,1.0)
```

but nevertheless the computed cameras are at least able to highlight the user's position with respect to some of the landmarks.

4.1 Performances

For each run, we have recorded the execution time (which includes both phases) and the quality of the generated camera, i.e. the best value of the fitness function. As it is shown in table 1, PSO is consistently faster than EXH, even when the grid resolution employed by EXH is ad-hoc set to minimize the number of considered points while still producing images with quality close to PSO. On the contrary, PSO parameters have not been changed across the problems. However, theoretically one could obtain better results by tuning the the size of the swarm depending on the number of properties defined. Note also that this result was not straightforward before doing the tests, since EXH works on a discretization of the search space, while PSO works in a continuous search space. With respect to the quality of the generated cameras, in the first two problems PSO obtains values of the fitness function close to the optimum (e.g. 4.87 out of 5 in the first problem), while in the last problem it is impossible to fully satisfy all requirements and therefore the degree of satisfaction is lower (28.73 out of 37.5).

As one can see from the table, one of the issues with PSO is the variance in the time needed. However, if there is the need of guaranteeing a solution within a certain amount of time, the algorithms has the nice property of being stoppable anytime and return a (possibly non optimal) solution.

From the results obtained, another evidence is that execution times generally depend on the number and nature of the properties that are evaluated with fitness functions, with occlusions, as noted in the literature, being the most costly evaluation.

Table 1. Performance data collected in 100 runs for PSO and EXH

	time (ms)					quality		
	PSO				EXH	PSO		EXH
Problem	min	max	average	std. dev.	average	best value	std. dev.	best value
over the shoulder (1.2)	33	319	151	36.24%	541	4.87/5.0	5.13%	4.84/5.0
occlusion (1.3)	58	737	443	36.95%	921	4.78/5.0	2.51%	4.80/5.0
navigation aid (1.4)	1762	3348	2235	16.63%	8993	28.73/37.5	6.41%	28.23/37.5

5 Conclusions

We have presented a PSO approach for the VCC problem. The proposed method works with static scenes and performs significantly better than an exhaustive search on a discretization of the space. Nevertheless, additional experiments on other scenes and image descriptions should be carried out to evaluate the method more thoroughly. With respect to this point, one goal for future work that could benefit the entire research area is to define a benchmark comprising a set of representative and realistic scenes and problems, thus allowing the experimental comparison of proposed methods in the literature.

Acknowledgments. The authors acknowledge the financial support of the Italian Ministry of Education, University and Research (MIUR) within the FIRB project number RBIN04M8S8.

References

[1] Bares, W., McDermott, S., Boudreaux, C., Thainimit, S.: Virtual 3d camera composition from frame constraints. In: Proceedings of the eighth ACM international conference on Multimedia, pp. 177–186. ACM, New York (2000)

[2] Bares, W.H., Gregoire, J.P., Lester, J.C.: Realtime constraint-based cinematography for complex interactive 3d worlds. In: AAAI/IAAI, pp. 1101–1106 (1998)

[3] Christie, M., Normand, J.-M.: A semantic space partitioning approach to virtual camera control. In: Proceedings of the Annual Eurographics Conference, pp. 247–256 (2005)

[4] Christie, M., Olivier, P.: Automatic camera control in computer graphics. In: Proceedings of the Annual Eurographics conference - State of the Art Reports, pp. 89–113 (2006)

[5] Eberhart, R.C., Kennedy, J.: Particle swarm optimization. In: Proceedings of the IEEE International Conference on Neural Networks 1995, vol. 4, pp. 1942–1948 (1995)

[6] Halper, N., Oliver, P.: CamPlan: A camera planning agent. In: Smart Graphics 2000 AAAI Spring Symposium, pp. 92–100 (2000)

[7] Kennedy, J., Eberhart, R.C., Shi, Y.: Swarm Intelligence. Morgan Kaufmann Publishers, San Francisco (2001)

[8] Pickering, J.H.: Intelligent Camera Planning for Computer Graphics. PhD thesis, Department of Computer Science, University of York (2002)

Through-the-Lens Scene Design

Ye Tao[1,2], Marc Christie[3], and Xueqing Li[1]

[1] Shandong University, School of Computer Science and Technology,
Ji Nan, 250100, P.R. China
[2] University of Nantes, Laboratoire d'Informatique de Nantes-Atlantique,
FRE CNRS 2729, 44300, Nantes, France
[3] IRISA/INRIA Rennes-Bretagne Atlantique,
Campus de Beaulieu, 35042, Rennes, France

Abstract. The increasing need in composing nice views of 3D scenes make it necessary to propose better editing tools that emphasize composition and aesthetics through high-level manipulators. In this paper, we propose to relieve the user from some low-level aspects of manipulation using a Through-the-lens declarative scheme. The projected features of the components that guide the composition of a picture (staging, camera and settings) are considered as primitives on which constraints can be applied. Specifically, we show how to compute the locations and orientations of objects, cameras and lights by constraining such features as vertex positions, specular reflections, shadows and penumbra.

1 Motivations

The increasing need in appealing views of 3D scenes makes it necessary to propose better editing tools that emphasize composition and aesthetics through high-level manipulators. Classical approaches such as those proposed by 3D modelers rely on a 4-view visualization of the scene, together with generic interaction metaphors for each entity. Lights, objects and cameras are moved, rotated and animated in the same way, by using translation arrows, arcballs and spline curve representations.

In a fundamental movement to re-design ways to interact with the content of the scenes, and following the work proposed in [1] and [2], we propose a Through-The-Lens declarative approach to improve the composition process that encompasses the on-screen specification of object projections, as well as camera and light manipulation. We offer specular light manipulation directly on the surface of the objects, as well as shadow and penumbra manipulation expressed in a unified framework.

2 State-of-the-Art

Easing the process of creating nicely composed shots is a target that has been long time explored both by graphics and HCI communities. Traditional 3D scene design, including adjusting the camera, constructing the objects, and positioning

A. Butz et al. (Eds.): SG 2008, LNCS 5166, pp. 142–153, 2008.

the lights to let the objects, highlights or shadows appear at desired positions on the screen, can be tedious work; the designer has to specify each parameter of every component, followed by a combination of various operations such as translation, rotation, and scaling, to attain the final desired visual image. Confronted to the difficulty in operationalizing and automating composition rules, researchers have proposed interactive and declarative approaches to assist users by directly specifying properties on the screen (screen-space constraints).

In early 80's, Jim Blinn [3] worked out the algebraic expression of positioning a camera from the input of two object locations on the screen. Gleicher and Witkin [1] proposed the Through-the-lens camera control technique, which allows the user to control a camera by setting constraints on object positions in the image plane, as seen through the lens. Therefrom, a number of approaches have employed this metaphor in interactive or declarative schemes for camera control. Declarative approaches to composition have been explored by [4,5] and [6]. Olivier et al. [5] express the problem in terms of properties to be satisfied on the screen (relative and absolute location, orientation, visibility), and use a stochastic search technique to compute possible solutions.

Extensions of this control metaphor have been proposed for manipulation of camera paths [7] and for very simple manipulations of objects and lights [2].

In this paper, we build upon and extend the work in [2] to propose a declarative approach to manipulate objects, camera and lights, directly through their properties on the screen. In the following section, we present an overview of the *Through-the-lens Cinematography* (TTLC) techniques and present how to express camera, object and light manipulation in a unified and consistent way.

3 Principles

Though the world is made of three-dimensional objects, in most cases, we do not communicate their properties directly, but their projections on a 2D image. The principle of the Through-the-lens Cinematography is to provide a user-guided interface that allows to interact with the 3D scene by manipulating its projection on the image plane.

Consider the n properties to be controlled, *e.g.* the position/orientation of an object/light/camera, as a n-dimension parameter vector q. The user's input, *i.e.* the desired target on screen, can be converted to a set of image-based constraints Q_i, where

$$Q_i : \{q \in \mathbb{R}^n | g_i(q) \circ 0\}, \circ \in \{<, \leq, =, \geq, >\}, i = 1, 2, \cdots, m. \tag{1}$$

Each constraint Q_i maps the parameter space ($q \in \mathbb{R}^n$) into a 2D image space ($\mathbb{R}^2$). Due to the high-dimensional nonlinear relations of g_i, we cannot guarantee that every instance of this problem has a closed-form solution. If there is no solution to q, which means that we cannot find parameters to satisfy all the constraints, we call it an over-constrained system. Otherwise, there are multiple solutions for q, which means that there are multiple ways to account for the changes to the image-space constraints, we call it an under-constrained system.

For example, if a user wants an object to be larger on screen, he can either move the camera closer or increase the zoom ratio (the value of the focal length) to achieve the same effect. In such cases, we need an objective function $F(q)$ to help to select the expected solutions from the universal set. It should be noticed that often several objectives must be traded off in some way, *e.g.* user may prefer to change the position of a camera rather than its focal length. The problem can then be represented as follows:

$$F(q) = \{f_1(q), f_2(q), \cdots, f_n(q)\}, \tag{2}$$

where $F(q)$ is a vector of all objectives.

The solution of q can then be found by solving a high-dimensional non-linear multiobjective optimization problem, defined as

$$\min_{q \in \mathbb{R}^n} F(q), s.t. q \in Q_i, i = 1, 2, \cdots, m. \tag{3}$$

Generally, there are two different types of constraint solver approaches, deterministic methods and stochastic methods. Jacobian solver interprets the original problem as an approximated linear problem, by formulating the relation between the screen velocities of a set of points $\dot{h}$ and the parameter vector velocity $\dot{q}$, as

$$\dot{h} = J\dot{q}, \tag{4}$$

where J is the combination of the Jacobian matrix of $g_i(q)$.

Gleicher and Witkin solved the matrix equation by considering the equation 4 as an optimization problem, which minimizes the variation of $\dot{q}$:

$$\min \frac{1}{2} \|\dot{q} - \dot{q}_0\|^2, s.t. J\dot{q} = \dot{h}_0. \tag{5}$$

Kyung *et al.* solved it by calculating the pseudo-inverse matrix of J:

$$\dot{q} = J^{-1}\dot{h}. \tag{6}$$

Above approaches generate the time derivatives of the *DOF*s, and convert the original non-linear problem to an approximated linear problem. The scene is updated iteratively using the velocity obtained from the linearized system. This metaphor usually provides smooth motion and interpolation, but requires an explicit Jacobian matrix derived from a velocities equation, which often results in a very complex formula. And in our experiment, the solution's accuracy is also reduced by converting the original non-linear problem as a linear approximation.

The Sequential Quadratic Programming (*SQP*) represents the state-of-the-art nonlinear programming method. By approximating the objective function as a quadratic function at each iteration, the algorithm intelligently chooses values for the variables that would put it at the minimum for that quadratic. It then updates the state with the actual evaluation and repeats this procedure until it has converged on a state that it views as the minimum.

In our experiments, both the Jacobian and the SQP solvers suffer strongly from the local minima problems, because both the objective function and the feasible area of the optimization problem are highly non-linear. We will present later the Genetic Algorithms (GA) to avoid the solver falling into the local minima.

This paper presents a unified declarative Through-the-lens 3D scene design metaphor, which integrates the camera, object and light design all together in an effective WYSIWYG interface. Instead of interacting with the parameters of every components, the user may now work on the final image and directly input some changes. The system models the user's inputs as constraints and objective functions, and automatically computes the parameters to satisfy the changes, by applying an optimization process. The scene is then be re-rendered according to the recomputed parameters.

In the following sections, we will detail the techniques for setting up the problem, including defining the objective functions and converting the user's input into constraints, before applying the solving techniques.

4 Camera Control

We start the introduction to the Through-the-lens technique by formulating the camera control as a constrained nonlinear problem. A virtual camera can be described by eleven DOFs (*i.e.* six extrinsic and five intrinsic properties). In most cases, however, there is no need to fully specify all the parameters. Here, we assume that the properties of the translation $tc = (tc_x, tc_y, tc_z) \in \mathbb{R}^3$, the unit quaternion rotation $qc = (qc_w, qc_x, qc_y, qc_z) \in \mathbb{R}^4$, the focal length $fc \in \mathbb{R}$ are able to change, and $c = (fc, tc, qc)$ is the composition vector of the camera parameters, while other properties are set to their default values.

We denote Vc_p as the nonlinear perspective projection of a point p from 3D space to 2D space, i.e.

$$Vc_p(c) = h(F_c(fc) \cdot T_c(tc) \cdot R_c(qc) \cdot p), \tag{7}$$

where F_c represents the perspective projection matrix, R_c and T_c represents the rotation and translation matrix between the world coordinates and the camera coordinates, respectively. $h(p)$ is the transformation that converts homogeneous coordinates into 2D coordinates.

The basic idea is to provide a set of virtual 2D points p'_i to control real projections of the 3D points $p_i, (i = 1, 2, \ldots, m)$. We set up the first m constraints by establishing the relations between p_i, which are already known in 3D models, and p'_i, which are given by the user, *i.e.*:

$$Q_i : p'_i = Vc_{p_i}(c), (i = 1, 2, \cdots, m). \tag{8}$$

In addition, the rotation quaternion must be normalized to unit magnitude, otherwise the object will be distorted. Instead of modifying the rotation matrix and normalizing it from time to time, we express the constraint as:

$$\|qc\| = \sqrt{qc_w^2 + qc_x^2 + qc_y^2 + qc_z^2} = 1. \tag{9}$$

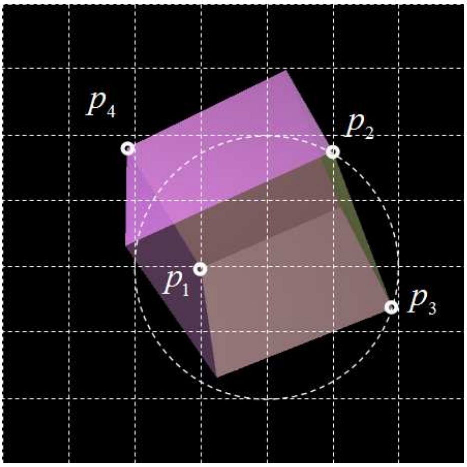


Fig. 1. Camera control with primitive and tolerance. $p'_1 = (0, 0)$, $\gamma : (x-0.1)^2+y^2 = 0.2^2$, $p'_4 = (-0.15, 0.15)$, $\delta_4 = 0.05$.

For the case of under-constrained problems, we expect a solution that minimizes the variation of c, which means that we want the camera to be adjusted as slightly as possible, the objective function can then be defined as:

$$\min \|c - c_0\|, \tag{10}$$

where c_0 is the initial guess value.

The simultaneous satisfaction of all the constraints is not always possible to obtain on rigid objects. Conflicts may exist between the constraints and weaker constraints can be applied to relax the image locations, for example, by introducing some tolerance. In these cases, constraints are presented as:

$$\|p' - Vc_p(c)\| < \delta, \tag{11}$$

where δ is the tolerance constant. In Fig. 1, p'_4 is constrained with a tolerant radius δ_4.

Each projected feature (generally a 2D point on the screen) can easily be constrained to a basic primitive as in [2]. Features are either fixed at a given position on the primitive or attached to a primitive (*i.e.* the feature can move freely on or inside the primitive).

4.1 Combination with Object Transformation

To make the scene design more flexible, the idea can be easily extended to object control. Let $to = (to_x, to_y, to_z) \in \mathbb{R}^3$ and $qo = (qo_w, qo_x, qo_y, qo_z) \in \mathbb{R}^4$, and T_o and R_o denote the classical translation matrix and quaternion rotation matrix respectively, then

$$Vo_p(o) = T_o(to) \cdot R_o(qo) \cdot p \tag{12}$$

defines how point p on an object O is transformed from local coordinates into world coordinates. It is possible to combine the camera and object manipulation together, by extending the parameter vector to $6N+7$ dimensions, for N objects manipulation, the constraints are then formulated as:

$$p\prime_i^j = Vc_{(Vo_{p_i^j}(o))}(c) = V_{p_i^j}(c, o), i = 1, 2, \ldots, m_j, j = 1, 2, \ldots, N, \tag{13}$$

where m_j is the number of the control points for the j^{th} object.

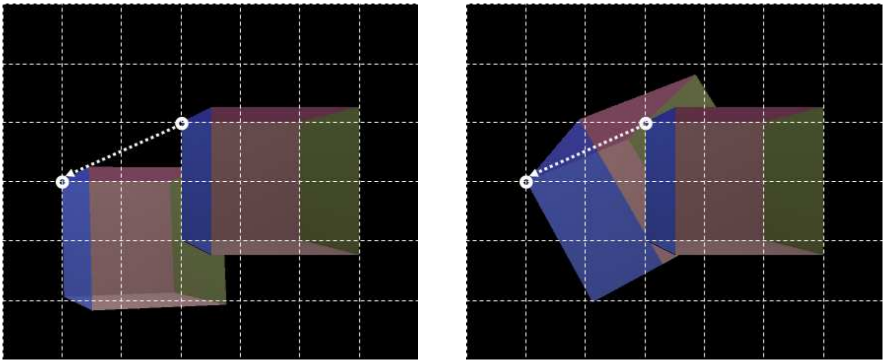
We want to minimize both the camera and object transformation variance, *i.e.* we want all the constraints to be satisfied while moving the object and the camera as slightly as possible, as shown in Fig. 2. This requires a multi-objective design strategy, which involves an expression of the objective function associated with both camera parameters, and the objects parameters.

The multi-objective vector F is converted into a scalar problem, by constructing a straightforward weighted sum of all the normalized objective functions, *i.e.*

$$F(q) = \min(\frac{\omega_c}{3} f_1(q) + \frac{1-\omega_c}{2N} \sum_{j=1}^{N} f_{j+1}(q)), \tag{14}$$

where ω_c is a proportional coefficient to balance the influence between the camera and object, f_1 models the objective related to the camera, and f_{j+1} the objective related to the objects.

Solving a constrained optimization problem with a large number of variables and highly nonlinear objective functions often makes the system performance poor. In our experiments, this process can be accelerated by reducing the number of iterations, and getting a near-optimal solution instead.



(a) Strong camera weighting in Equation. 14

(b) Strong object weighting in Equation. 14

Fig. 2. Camera and object manipulation with a balacing weighting

5 Specular Reflection Control

Lighting design consists in finding a group of appropriate settings of the camera, objects and light source that result in a desired illumination effect, such as the highlights and shadows. This section focuses on specular reflection design, and the next section will be dedicated to shadow design.

We start the light control by using a single point light source to model the hightlight from a surface as having a specular reflection. An omni-directional point light, whose rays diverge in all directions, can be modeled by its position $l = (tl_x, tl_y, tl_z)$.

For complex surfaces, the expression of the surface can be locally approximated by a 3D plane. Let us consider a plane $T : Ax + By + Cz + D = 0$ in 3D space that contains the triangular face (v_1, v_2, v_3) with geometric center v_c, as shown in Fig. 3(a).

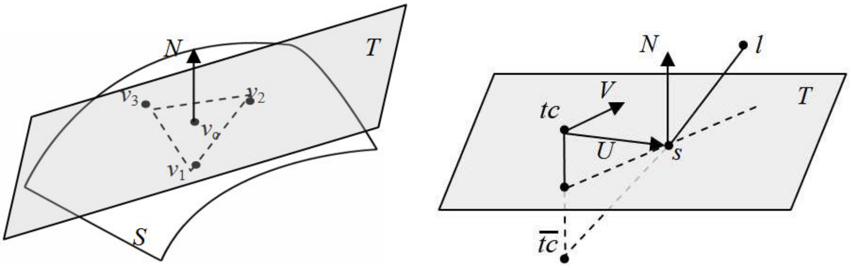
T can be calculated by (v_1, v_2, v_3), where

$$N = [A\ B\ C]' = \frac{(v_2 - v_1) \times (v_3 - v_2)}{\|(v_2 - v_1) \times (v_3 - v_2)\|}, \tag{15}$$

and

$$D = -v_1 \cdot N. \tag{16}$$

Knowing T, s can be found straightforwardly by intersecting T and $l\bar{tc}$, where $\bar{tc}$ is the mirror of tc about T. We define $U = s - tc$, and V is the view direction, as shown in Fig. 3(b).



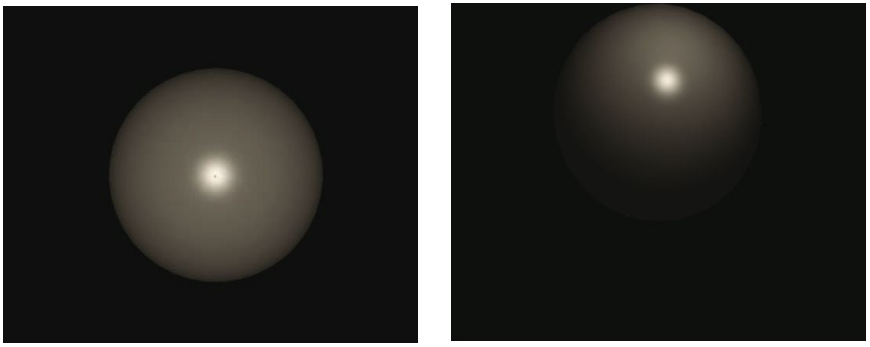
(a)

(b)

Fig. 3. Computation of the specular reflection position

As discussed in the previous sections, the proposed TTLC interface allows to specify the reflection point directly on screen. We cast a ray from the view point through the user-defined reflection point on the image plane, to detect the triangular face on which the specular reflection point should be, and apply the above algorithms to obtain the expression of s. The constraint can then be expressed as the projection of s in image space.

$$p' = Vc_s(c). \tag{17}$$



(a) (b)

Fig. 4. The specular reflection control

Some additional constraints must be involved to avoid unreasonable solutions (such as setting the light opposite to the camera, and avoiding the light being too closely located to the specular reflection).

The position of the specular reflection depends on the relative placement between the object, the camera and the light source. It is clear that we need to model a multi-objective optimization problem to help to find a proper placement of the three components, which is similar to the combination of object and camera design. Fig. 4(a) and Fig. 4(b) illustrate a unit sphere with the specular reflection position set to $(0, 0)$ and $(0, 0.5)$.

6 Shadow Control

The shading in a scene depends on a combination of many factors. We present an ellipse-based approach to solve the inverse shading problem, in which the desired shadow effects, including the positions of the shadows, and umbra/penumbra areas, are all modelled by ellipses.

6.1 Umbra Design

Now, consider the process of positioning one point light by specifying the bounding ellipse of the shadow. The abstracted question is stated as: Given an object O, a plane P, and a bounding ellipse E_0 on P, find a point light source positioned at l, such that the generated shadow best matches E_0.

Suppose that O is defined by n vertices, *i.e.* $v_1, v_2, \cdots\cdots, v_n$. We first compute the Minimum Volume Enclosing Ellipsoid (MVEE) of O, denoted as

$$M = \left\{x \in \mathbb{R}^3, (x - c)^T Q_C (x - c) \leq 1\right\}, \tag{18}$$

where c is the center of M, and Q_C is a 3×3 symmetric positive definite matrix, which can be found by solving the optimization problem:

$$\begin{array}{cc} \min & \log\left(\det\left(Q_C\right)\right) \\ s.t. & (v_i - c)^T Q_C (v_i - c) \leq 1, \\ & i = 1, 2, \ldots, n. \end{array} \tag{19}$$

To simplify the expression, we denote Q_M a homogeneous 4×4 matrix of M, such that

$$Q_M = T_C^T \begin{pmatrix} Q_C & 0 \\ 0 & 1 \end{pmatrix} T_C, \tag{20}$$

where

$$T_C = \begin{pmatrix} 1 & 0 & 0 & -c_x \\ 0 & 1 & 0 & -c_y \\ 0 & 0 & 1 & -c_z \\ 0 & 0 & 0 & 1 \end{pmatrix}. \tag{21}$$

We now try to find the silhouette E on P casted by M from l, *i.e.* the intersection of tangent cone S to M passing through l, and the plane P, as shown in Fig. 5(a). S is given by the set of points $m : (x, y, z)$ such that the line determined by lm is tangent to M. Let t be a real number, such that $l + t(l - m)$ is on M, namely given by

$$S := (l + t(l - m))^T Q_M (l + t(l - m)) = 0. \tag{22}$$

Consider Equation (22) as a quadratic equation with only one unknown variable t. The line lm, as defined, is tangent to E, thus, there is only one point of intersection of lm and E, which implies that there is one and only one t satisfying Equation (22). We then can assert that Equation (22) has the unique solution, *i.e.*

$$\Delta_t = f(x, y, z, l, Q_M) = 0, \tag{23}$$

where Δ_t is the discriminant as regards to t. Knowing l and Q_M, we then obtain the expression of S from Equation (23), which can be rewritten in a homogeneous form

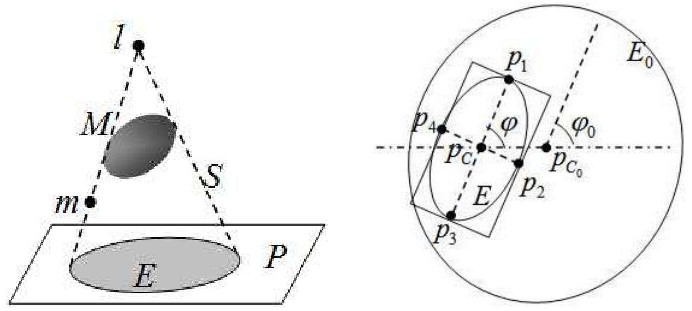
$$S = \left\{ x \in \mathbb{R}^3, \begin{pmatrix} x^T & 1 \end{pmatrix} Q_S \begin{pmatrix} x \\ 1 \end{pmatrix} = 0 \right\}, \tag{24}$$

where Q_S is a 4×4 symmetric positive definite matrix.

Plane P can be transformed to XY-plane by a standard rotation matrix R_P and a translation matrix T_P, whose elements are derived from the expression of P. We apply the identical transformation matrices on Q_S, and obtain

$$\widehat{Q_S} = T_P^T R_P^T Q_S R_P T_P. \tag{25}$$

Finally, this enables us to get the 2D equation of E by simply removing the third row and column in $\widehat{Q_S}$. E is required to be as close as possible to E_0. This can be achieved by maximizing the area of E while maintaining E in E_0. The area of E can be easily calculated by $\pi \cdot a \cdot b$, where a and b are the semi-axes of E. The constraint that E is enclosed in E_0 can be satisfied when all the four tangent points $p_i, i = 1, 2, 3, 4$ on the bounding rectangle are enclosed in E_0, as shown in Fig. 5(b).



(a) Calculation of the silhouette E

(b) The relative position of E_0 and E

Fig. 5. Shadow design with point light

Now we model the constrained optimization problem as follows:

$$\begin{array}{ll} \min & -a^2b^2 \\ s.t. & p_i^TQ_{E_0}p_i \le 0, i = 1, 2, 3, 4. \end{array} \tag{26}$$

A less expensive way to express the constraints is to consider the center p_C and the rotation angle φ of E. Locate p_C at p_{C_0}, and let φ be equal to φ_0, where p_{C_0} and φ_0 are the center and the rotation angle of E_0 respectively. In this case, if a and b are shorter than the semi-axes of E_0, denoted by a_0 and b_0, we say that E is guaranteed to be enclosed in E_0. This can be written in the following equality and inequality constraints:

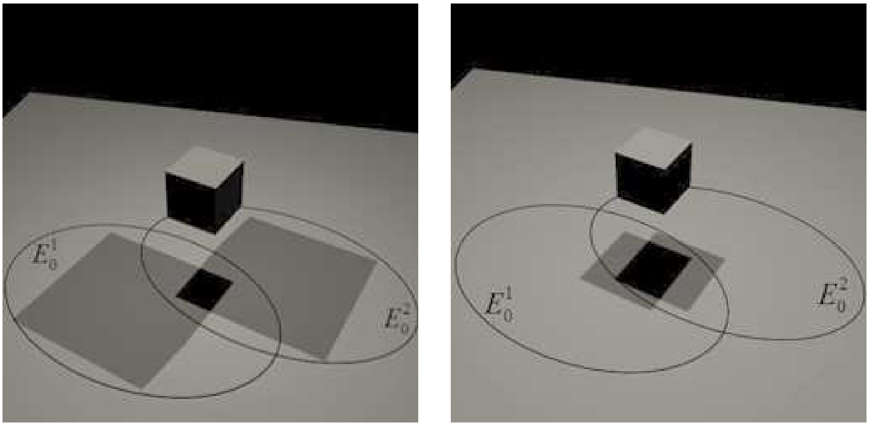
$$p_C - p_{C_0} = 0, \varphi - \varphi_0 = 0, a - a_0 < 0, b - b_0 < 0. \tag{27}$$

Note that the objective function and the constraints in Equation (26) contain only one unknown vector $l = (l_x, l_y, l_z)^T$. Therefore, the position of the point light can be computed by using a non-linear constraint solver.

The proposed algorithm can be extended to design shadows for multiple objects and complex objects, which can be divided to several parts, $O_1, O_2, \cdots\cdots, O_m$. And we apply the same techniques to calculate the shadow ellipse E_i for each O_i, $i = 1, 2, \cdots\cdots, m$ respectively, and find the position of the point light by solving a similar optimization problem.

6.2 Penumbra Design

If there are multiple light sources, there are multiple shadows, with overlapping parts darker. For a linear light source, the generated shadow is divided into the umbra area and penumbra area. Suppose for the moment that a linear light is restricted to its simplest case where only two point lights are located at the vertices l_1 and l_2. As discussed earlier, the user specifies the positions of the ellipses to indicate where the shadow should be located on the plane, as shown in Fig. 6(b).



(a) Two free point lights

(b) Linear light

Fig. 6. Penumbra design with ellipses

The umbra is specified by intersecting E_0^1 and E_0^2, while the penumbra is specified by the rest of the area in the two ellipses, *i.e.*

$$\begin{array}{rl} \left(E_0^1 \cap E_0^2\right) & \textit{umbra area}, \\ \left(E_0^1 \cup E_0^2\right) - \left(E_0^1 \cap E_0^2\right) & \textit{penumbra area}. \end{array}$$

We extend the previous point light algorithm described in [2] to two vectors l_1 and l_2. If we consider two *individual* point lights, we can make the penumbra area fit the ellipses to the maximum extent.

Then, for a linear light, we add an additional constraint to guarantee that the scope of the light is unchanged in the 3D space, *i.e.*

$$\|l_1' - l_2'\| = \|l_1 - l_2\| \, , \tag{28}$$

which means that the distance between l_1' and l_2' is a fixed value. The solving techniques are identical to the one-point-light design. The umbra in Fig. 6(b) successfully fits the intersection area as we expected, while the penumbra, however, cannot fully fit the union area of the two ellipses, due to the rigid properties of the linear lights.

7 Discussion

This paper has presented a declarative approach to scene design that includes object, camera and light manipulation. It extends the previous approaches and offers a unified solving approach based on non-linear constraint solvers. Due to the complexity of the problems and the size of the search spaces, the computational costs remain important. To offer real-time interactive manipulations will require an abstraction of the models and a simplification of the relations, possibly at the cost of precision. Our next step will be to move this cursor between cost and precision while maintaining the expressive power of the interaction in proposing a new generation of manipulation tools for easing screen composition.

References

1. Gleicher, M., Witkin, A.: Through-the-lens camera control. In: SIGGRAPH 1992: Proceedings of the 19th annual conference on Computer graphics and interactive techniques, pp. 331–340. ACM, New York (1992)
2. Christie, M., Hosobe, H.: Through-the-lens cinematography. In: Butz, A., Fisher, B., Krüger, A., Olivier, P. (eds.) SG 2006. LNCS, vol. 4073, pp. 147–159. Springer, Heidelberg (2006)
3. Blinn, J.: Where am i? what am i looking at? [cinematography]. Computer Graphics and Applications, IEEE 8(4), 76–81 (1988)
4. Bares, W., McDermott, S., Bouderaux, C., Thainimit, S.: Virtual 3D camera composition from frame constraints. In: Proceedings of the 8th International ACM Conference on Multimedia (Multimedia 2000), pp. 177–186. ACM Press, New York (2000)
5. Olivier, P., Halper, N., Pickering, J., Luna, P.: Visual Composition as Optimisation. In: AISB Symposium on AI and Creativity in Entertainment and Visual Art, pp. 22–30 (1999)
6. Christie, M., Languénou, E.: A constraint-based approach to camera path planning. In: Butz, A., krüger, A., Olivier, P. (eds.) SG 2003. LNCS, vol. 2733, pp. 172–181. Springer, Heidelberg (2003)
7. Barrett, L., Grimm, C.: Smooth key-framing using the image plane. Technical Report 2006-28, Washington University St. Louis (2006)

Similarity-Based Exploded Views

Marc Ruiz[1], Ivan Viola[2], Imma Boada[1], Stefan Bruckner[3], Miquel Feixas[1], and Mateu Sbert[1]

[1] Graphics and Imaging Laboratory, University of Girona, Spain
[2] Department of Informatics, University of Bergen, Norway
[3] Institute of Computer Graphics and Algorithms,
Vienna University of Technology, Austria

Abstract. Exploded views are often used in illustration to overcome the problem of occlusion when depicting complex structures. In this paper, we propose a volume visualization technique inspired by exploded views that partitions the volume into a number of parallel slabs and shows them apart from each other. The thickness of slabs is driven by the similarity between partitions. We use an information-theoretic technique for the generation of exploded views. First, the algorithm identifies the viewpoint which gives the most structured view of the data. Then, the partition of the volume into the most informative slabs for exploding is obtained using two complementary similarity-based strategies. The number of slabs and the similarity parameter are freely adjustable by the user.

1 Introduction

Volume visualization aims at gaining insight into volumetric data using interactive graphics and imaging techniques. Current volume data sets generated by scientific domains contain large amounts of data of complex structures. Effective visualization of such data sets that clearly shows all contained structures is challenging.

Illustrative visualization enhances the expressiveness of volume rendering by applying hand-crafted illustrative techniques. Cut-aways, exploded views or high-level abstraction strategies, amongst others, are used to reveal insights and represent essential structures of the volume in a clear way while less important details are subjugated. To employ these techniques, certain controlling mechanisms based on data or higher semantical levels (e.g. segmentation into objects from the domain perspective and the assigning of object importance based on the given domain scenario) are required. These mechanisms vary from fully interactive steered by user (e.g. voxel-by-voxel segmentation) to fully automatic techniques (e.g. shape analysis of the acquired data based on higher-order derivatives). To explore unclassified data sets, automatic controlling mechanisms for steering expressive visualization are useful, and possibly can be combined with interactive techniques that *fine-tune* the first automatic *educated guess*.

Our interest is focused on exploded views, which partition the volume into different parts that are displaced away from each other as if there had been a small

A. Butz et al. (Eds.): SG 2008, LNCS 5166, pp. 154–165, 2008.

controlled explosion emanating from the focus of interest. Exploded views enable to see details of otherwise overlapping structures, exploiting the observer's understanding of the original spatial arrangement. In this paper, a new partitioning approach for automatic generation of exploded views is presented. This method divides the data set into a set of slabs defined by parallel planes, combining in this way the advantages of 2D and 3D views. While 3D visualization provides a global view of the entire model, the 2D cross sectional views reveal insights. To partition the volume, two alternative strategies are proposed. The first one starts with the entire volume and partitions it recursively guided by a *maximum dissimilarity* criterion. The second one considers initially all individual slices and groups them together according to a *similarity* criterion. In both cases, the controlling mechanism is the similarity value that is computed automatically using information-theoretic measures. The only necessary interaction of the user with the data is a single threshold parameter which determines when the partitioning (or grouping) has to stop. An important advantage of this approach is that no a-priori information or pre-processing of the data is required. This is suitable, especially, for computer-guided exploration of histology volume data.

2 Related Work

The main limiting factor when exploring volume data is the occlusion between structures. For complex volumetric data sets it is difficult to achieve a visual representation that not only shows all the internal structures but also preserves the global representation of the model. To enhance volume data interpretation Rheingans and Ebert [1] introduced the volume illustration approach, combining the familiarity of a physics-approximated illumination model with the ability to enhance important features using non-photorealistic rendering techniques. Volume illustration techniques enhance the perception of structure, shape, orientation, and depth relationships in a volume model. Although they cannot totally solve the occlusion problem, the good performance of these techniques led to the development of new volume rendering methods.

Clipping away or removing away parts of the data to eliminate occlusion is a well-known and extensively used approach. The loss of context due to removed parts is the main limiting factor of such a technique. To overcome this limitation, strategies with more complex clipping geometry have been proposed. Wang et al. [2] introduced volume sculpting as a flexible approach to explore data. Weiskopf et al. [3] proposed several interactive clipping techniques that are capable of using complex clip geometries. Konrad-Verse et al. [4] described a method which is based on a deformable cutting plane for virtual resection. Viola et al. [5] presented an importance-driven approach capable of enhancing important features while preserving the necessary context by generating cut-away views and ghosted images from volumetric data. Bruckner et al. [6] proposed an alternative to conventional clipping techniques in order to avoid loss of context. Their context-preserving volume rendering model uses a function of shading intensity, gradient magnitude, distance to the eye point, and previously accumulated opacity to selectively reduce the opacity in less important data regions.

Exploded views and deformations are a common strategy for communicating the structure of complex 3D objects that are composed of many subparts. Deformation metaphors for browsing structures in volumetric data were introduced in volume visualization by McGuffin et al. [7]. They presented an approach for volume exploration based on deformations that allows the users to cut into and open up, spread apart, or peel-away layers of the volume while still retaining the surrounding context. The explosion of the parts is set manually. Bruckner et al. [6] went one step further by automating the explosion. Their method uses a continuous degree-of-interest function to distinguish between focus and context and is capable of re-arranging the parts dynamically based on the viewpoint. In these techniques, a priori knowledge of the volume data to define the layers or to set the focus of interest is assumed – the data has been explicitly partitioned by the user. In contrast, our approach automatically partitions the volume based on characteristics of the data.

Moreover, good viewpoint selection is also crucial for an effective focus of attention [5]. Different information-theoretic measures for viewpoint evaluation have been presented. Vàzquez et al. [8] have introduced the viewpoint entropy as a measure for viewpoint quality evaluation, where the best viewpoint is defined as the one that has maximum entropy. Designed for polygonal data, this measure has been extended to volumetric scalar data [9,10]. Viola et al. [11] have presented the viewpoint mutual information from the definition of an information channel between a set of viewpoints and a set of objects of a volumetric data set. This measure provides representative views and is very robust with respect to the resolution of the model.

3 Similarity-Steered Visualization

To automatically obtain the partitioning planes for the exploded views, we propose a two-step process. First, we select the view of the model along which the organs or components will be better separated. This view is called the *most structured view* of the model. Second, we calculate the partitions of the model along the most structured view. Such partitions will be obtained using two complementary approaches: a top-down strategy that divides the model according to the maximum information gain and a bottom-up method that joins the slices according to a similarity criterion. Then, the explosion of the model is visualized in the interactive system VolumeShop [12]. The two steps of the method are described below.

1. **Selection of splitting axis**
   The goal of this step is to obtain the most structured view of the model. To reach this objective a viewpoint measure able to capture the structure of the volumetric dataset along any view axis is used. In information theory, *entropy rate* is defined as a measure of the irreducible randomness of an object or the degree of unpredictability of a sequence of values. Since a high randomness corresponds to a low structure and vice versa, we can use the entropy rate

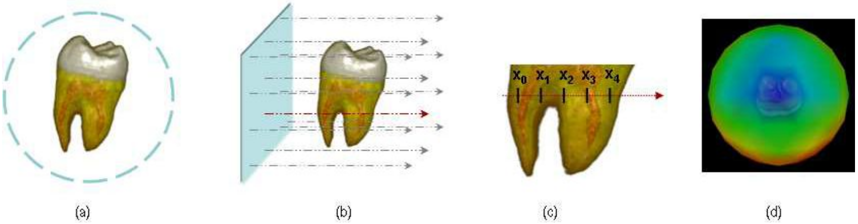


Fig. 1. Main steps of the selection of the most structured view. (a) Sphere of viewpoints, (b) sampling process for one viewpoint, (c) samples considered for the entropy rate computation, and (d) colored viewpoint sphere (using a thermic scale, from blue to red) representing the values of the viewpoint entropy rate.

to quantify the degree of structure or predictability of a model. We proceed as illustrated in Figure 1. First of all, the model is centered in a viewpoint sphere built from the recursive discretisation of an icosahedron (Figure 1(a)). Then, for each viewpoint the entropy rate is computed as described in Section 4 (Figure 1(b,c)). Finally, we identify the lowest entropy rate value which corresponds to the most structured view of the model (Figure 1(d)). This direction is used as axis to which similarity-based partitioning planes are perpendicular to.

2. **Volume partitioning**
   This task consists of selecting the optimal partitions of the model from the most structured view. To carry out this process two different strategies are presented:
   (a) **Top-down approach.** Initially, the entire volume is considered and partitioning planes are taken perpendicular to the most structured view (Figure 2(i.a)). To divide the dataset into different parts, we use a greedy algorithm which successively selects the partition that provides us with the maximum gain of information. According to the information bottleneck method [13,14], the information gain can be calculated using the Jensen-Shannon divergence between two parts of the model (Figure 2(i.b)). This measure can be interpreted as the degree of dissimilarity between the parts and attempts to divide the model into homogeneous regions (Figure 2(i.c)). A more detailed description of this approach is given in Section 5.1.
   (b) **Bottom-up approach.** All the slices of the volume, perpendicular to the most structured view, are considered as the initial slabs (Figure 2(ii.a)). Neighboring slabs are iteratively grouped (Figure 2(ii.b)) when mutual information between them is higher than a given threshold (Figure 2(ii.c)). Dealing with similarity between slabs instead of individual slices, we avoid an incorrect grouping, for instance, due to smooth changes along many consecutive slices. The grouping process is further described in Section 5.2.

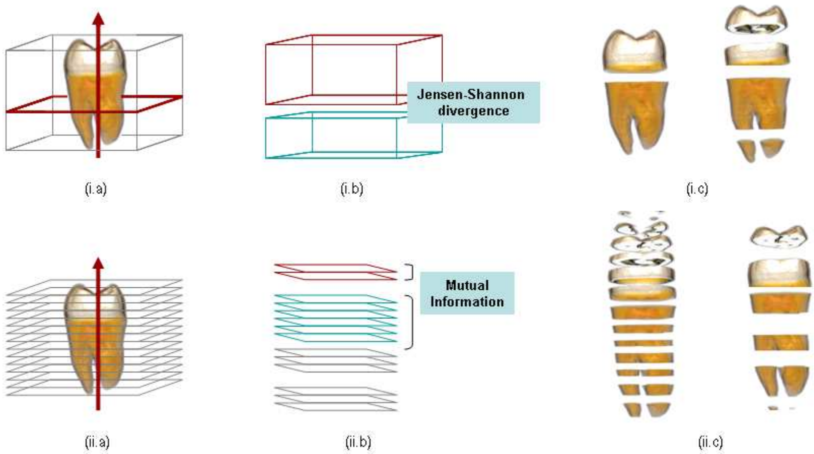


Fig. 2. (i) Top-down volume partition: (i.a) partitioning planes are taken perpendicular to the most structured view, (i.b) dissimilarity between subvolumes is given by the Jensen-Shannon divergence, and (i.c) examples showing two different partitions. (ii) Bottom-up volume partition: (ii.a) slices are taken perpendicular to the most structured view direction, (ii.b) similarity between slices or slabs is computed using mutual information, and (ii.c) two examples resulting from the grouping process.

4 Selection of Structured Views

To quantify the degree of structure of a volumetric data set along a given viewing direction, we estimate the entropy rate of the sequence of values (intensities) obtained by casting a bundle of parallel lines along that direction. These lines act as probes to sample the intensity of the model. The view with the lowest entropy rate will correspond to the most structured view.

The definitions of both Shannon entropy and entropy rate [15] are now reviewed. The notation used is inspired by the work of Feldman and Crutchfield [16]. Let $\mathcal{X}$ be a finite set and X a random variable taking values x in $\mathcal{X}$ with probability distribution $p(x) = Pr[X = x]$. The *Shannon entropy* $H(X)$ of a random variable X is defined by

$$H(X) = -\sum_{x \in \mathcal{X}} p(x) \log p(x). \tag{1}$$

The Shannon entropy measures the average uncertainty of random variable X. If the logarithms are taken in base 2, entropy is expressed in bits.

Given a sequence $X_1 X_2 \ldots$ of random variables X_i taking values in $\mathcal{X}$, a block of L consecutive random variables is denoted by $X^L = X_1 \ldots X_L$. The probability that the particular L-block x^L occurs is denoted by joint probability $p(x^L) = p(x_{i+1}, \ldots, x_{i+L})$. The *joint entropy* of a block of L consecutive symbols or L-*block entropy* is defined by

$$H(X^L) = -\sum_{x^L \in \mathcal{X}^L} p(x^L) \log p(x^L), \tag{2}$$

where the sum runs over all possible L-blocks.

The *entropy rate* or *entropy density* is defined by

$$h = \lim_{L \to \infty} \frac{H(X^L)}{L} \tag{3}$$

and measures the average amount of information per symbol x [15].

It can also be rewritten as

$$h = \lim_{L \to \infty} (H(X^L) - H(X^{L-1})). \tag{4}$$

The entropy rate of a sequence of symbols is a measure of its uncertainty, randomness or unpredictability. The entropy rate is also a measure of the compressibility of a sequence: the higher the uncertainty, the lower the compressibility. For instance, in a text, if there are strong correlations between letters (or words), knowledge of all previous letters (or words) will significantly decrease our uncertainty about the next one [16].

How to compute the entropy rate for a given viewpoint is now shown. Consider the scheme in Figure 1. For each viewpoint, a sequence of samples (intensity values) to compute the measure is obtained performing a ray casting from a plane centered at the viewpoint. We proceed as follows:

- From the plane at each viewpoint, parallel rays with a regular horizontal and vertical spacing x are cast. Along the ray within the volume, equidistant samples at distance y are taken.
- To build the two joint histograms of X^L and X^{L-1} required for the entropy rate computation, we take into account all possible groups of consecutive samples of length L and $L-1$, respectively. For example, with the samples shown in Figure 1(c), we can form three blocks of length 3 ($x_0x_1x_2$, $x_1x_2x_3$ and $x_2x_3x_4$) and four of length 2 (x_0x_1, x_1x_2, x_2x_3 and x_3x_4) for X^3 and X^2 histograms, respectively.
- From the joint histograms of X^L and X^{L-1}, the joint probability distributions $p(x^L)$ and $p(x^{L-1})$ are estimated and then the joint entropies $H(X^L)$ and $H(X^{L-1})$ are calculated.
- Due to the potentially high dimensionality of the histograms and, consequently, the high number of components, a trade-off between the number of symbols (intensities) and the length L of the blocks has to be considered. Note that the size (number of entries) of the highest histogram is $O(N^L)$, where N is the number of different property values and L is the length of the blocks. Usually voxel models have property values of 8 bits or more, so this problem is untreatable even with short blocks. As entropy rate is a limit quantity (4), its computation would require an infinite number of elements and blocks infinitely long. It has to be approximated using a block of finite length. From the two possible approximations coming from (4) and (4), we

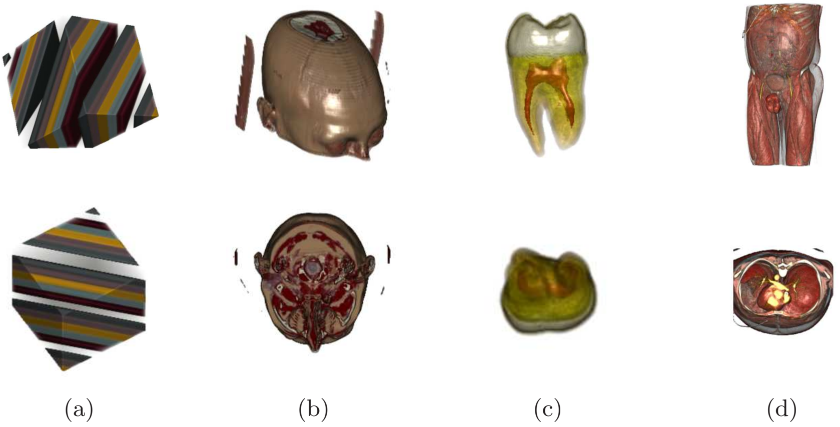


Fig. 3. Different volumes (first row) and their corresponding most structured views (second row). From left to right: (a) a synthetic model, (b) a CT-scan of a patient with hemorrhage, (c) a CT-scan of a tooth and (d) a CT-scan of the human body.

have selected the last one because it approximates more accurately the entropy rate for low values of L. In our experiments, we have taken $L = 3$. We have also reduced the number of symbols of $\mathcal{X}$ intensity bins in the histogram to 32, rescaling the intensity bins.

The strategy for the selection of the most structured view has been applied to different volume data sets. The obtained results are illustrated in Figure 3 where the first row represents the original model and the second row the most structured view. From left to right, the proposed models correspond to: (a) a synthetic model, created considering six different materials (each one represented with a different color) which follow a diagonal distribution through the volume, (b) a CT-scan of a patient with an hemorrhage, (c) a CT-scan of a tooth and (d) a CT-scan of the human body. Observe how the best views show the maximum of structure in the model. This is specially noticeable in the phantom model (Figure 3(a)) where the different regions have an inclination relative to the cube axis.

5 Evaluating Similarity

To obtain the optimal partitions for the explosion of a 3D data set, two different strategies are presented. First, we analyze a top-down approach which partitions the model using a criterion of maximum gain of information. Second, we study a bottom-up strategy that groups the slices according to a similarity measure between them.

5.1 Model Partitioning

Once the most structured direction of the model has been selected, a sequence of perpendicular partitions in that direction can be obtained using a simple greedy algorithm. This is a top-down hierarchical application of the information bottleneck method [13,14] which permits us to measure the gain of information when a model is divided into different slabs. This *gain of information* is computed using the Jensen-Shannon divergence.

The *Jensen-Shannon divergence* [17] between probability distributions p_1, p_2, ..., p_N with prior probabilities or weights $\pi_1, \pi_2, \dots, \pi_N$ is defined by

$$JS(\pi_1, \pi_2, \dots, \pi_N; p_1, p_2, \dots, p_N) \equiv H\left(\sum_{i=1}^{N} \pi_i p_i\right) - \sum_{i=1}^{N} \pi_i H(p_i), \qquad (5)$$

where $\sum_{i=1}^{N} \pi_i = 1$. The JS-divergence measures how far the probabilities p_i are from their likely joint source $\sum_{i=1}^{N} \pi_i p_i$ and equals zero if and only if all the p_i are equal. From [14], it can be seen that the gain in information when a dataset is divided into two slabs is given by

$$\mathit{\Delta} I = \frac{v_1 + v_2}{v_T} JS(\frac{v_1}{v_1 + v_2}, \frac{v_2}{v_1 + v_2}; p_1, p_2), \qquad (6)$$

where v_1 and v_2 are, respectively, the volumes of slabs 1 and 2, v_T is the total volume of the 3D dataset, p_1 and p_2 are, respectively, the normalized intensity histograms of slabs 1 and 2, and $JS(\frac{v_1}{v_1+v_2}, \frac{v_2}{v_1+v_2}; p_1, p_2)$ is the Jensen-Shannon divergence between p_1 and p_2 with the corresponding weights $\frac{v_1}{v_1+v_2}$ and $\frac{v_2}{v_1+v_2}$.

The gain of information when a model is divided into two parts is given by the dissimilarity between them (measured by JS) weighted by their relative volume. Note that a slab highly structured along a given direction will have all possible partitions very similar and thus will not need to be partitioned.

5.2 Slice Grouping

Given a viewing direction, the slices perpendicular to it can be grouped using a similarity measure. In this paper, the normalized mutual information is used to quantify the degree of similarity between individual slices or groups of adjacent slices (slabs). In medical imaging, many successful automatic image registration methods are based on the maximization of mutual information. This method, introduced by Viola [18] and Maes et al. [19], is based on the conjecture that the correct registration corresponds to the maximum mutual information between the overlap areas of the two images. Later, Studholme et al. [20] proposed to use the normalized mutual information as it is more robust than solely mutual information due to its greater independence of the overlap area.

Let X and Y be two random variables taking values x and y in finite sets $\mathcal{X}$ and $\mathcal{Y}$ with probability distributions $p(x) = Pr[X = x]$ and $p(y) = Pr[Y = y]$, respectively. The *mutual information* between X and Y is defined by

$$MI(X, Y) = \sum_{x \in \mathcal{X}} \sum_{y \in \mathcal{Y}} p(x, y) \log \frac{p(x, y)}{p(x)p(y)}, \qquad (7)$$

where $p(x, y) = Pr[X = x, Y = y]$ is the joint probability. MI is a measure of the shared information or the degree of dependence between X and Y. MI is zero only if the two random variables are strictly independent. The *normalized mutual information* is defined by

$$NMI(X,Y) = \frac{MI(X,Y)}{H(X,Y)}, \tag{8}$$

where $H(X,Y)$ is the joint entropy of X and Y. NMI takes values in the range [0,1].

An explanation is now given to compute the NMI measure between slices and the algorithm to group them. Given two slices A and B from the volume dataset, with associated random variables X and Y, the joint probability distribution $p(x,y)$ can be estimated by simple normalization of the joint histogram $h(x,y)$ of both slices. This is obtained from the intensities of each pair (a,b) of corresponding voxels, where $a \in A$ and $b \in B$. Once the joint histogram has been calculated, the joint probability distribution and the marginal probability distributions of X and Y can be estimated: $p(x,y) = \frac{h(x,y)}{\sum_{x \in \mathcal{X}} \sum_{y \in \mathcal{Y}} h(x,y)}$, $p(x) = \sum_{y \in \mathcal{Y}} h(x,y)$ and $p(y) = \sum_{x \in \mathcal{X}} h(x,y)$. The similarity measure NMI is then evaluated.

The similarity between two slices can be extended to the similarity between two slabs $\widehat{A} = \{A_1, \ldots, A_n\}$ and $\widehat{B} = \{B_1, \ldots, B_m\}$. The random variables $\widehat{X}$ and $\widehat{Y}$, associated with both slabs, represent the grouping of a set of random variables $\{X_1, \ldots, X_n\}$ and $\{Y_1, \ldots, Y_m\}$, respectively. Their joint frequency histogram is obtained from the intensities of each pair of corresponding voxels (a_i, b_j), where $a_i \in A_i$ and $b_j \in B_j$ $\forall i, j$. As mentioned above, the joint and marginal probability distributions can be estimated and thus the NMI measure is obtained.

Given the similarity measure NMI, the algorithm proceeds by joining the two adjacent slabs with maximum similarity. This process stops when the similarity between them is above a user-defined threshold or a number of slabs has been reached. At the beginning, every slab consists of only one slice. Then, the most similar slabs are progressively joined. To group n slices, the algorithm proceeds as follows:

- Assign n slabs such that each slab contains exactly one slice.
- Compute NMI for each pair of consecutive slabs.
- Find the two closest consecutive slabs i and $i+1$ (with maximum NMI). If the similarity between them is higher than the given threshold, then create a new slab $\widehat{i}$ by combining i and $i+1$ and recalculate NMI for the neighboring slabs of $\widehat{i}$. This step stops when the similarity between each pair of consecutive slabs is lower than a fixed threshold or a number of slabs is achieved.

5.3 Results

These proposed approaches have been implemented and integrated into the VolumeShop framework [12]. To test the methods, different synthetic and real data sets have been considered. In all the tests a sphere of 42 viewpoints has been

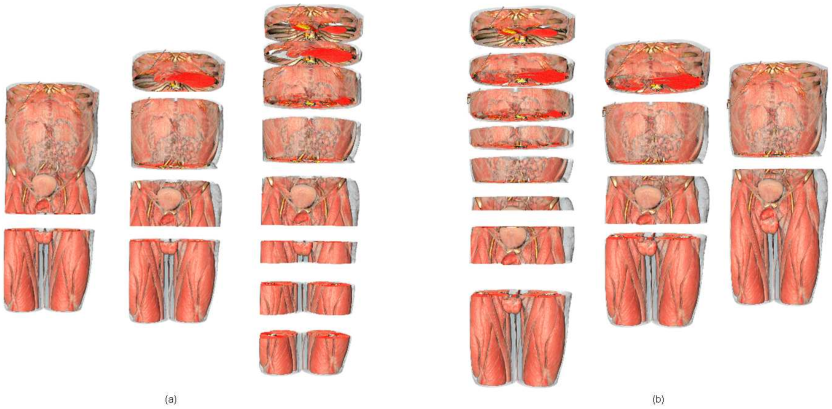


Fig. 4. (a) The volume partitioning and (b) the slice grouping approaches applied to a CT scan of the human body

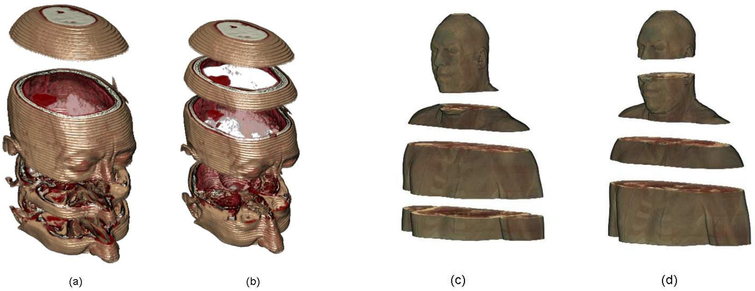


Fig. 5. (a) The volume partitioning and (b) the slice grouping approaches applied to a CT-scan of a patient with a brain hemorrhage. An histological data model decomposed with the (c) volume partitioning and (d) slice grouping methods.

used and the stopping criterion has been fixed by the number of slabs entered by the user.

For the first tests, the CT scan of the human body of Figure 4 has been used. The results obtained with the model partitioning approach for 2, 4 and 8 partitions are illustrated in Figure 4(a). In Figure 4(b) the partitions obtained with the slice grouping approach using the same user parameters are shown. In Figures 5(a) and (b) we illustrate the results obtained by applying the volume partitioning and the slice grouping approaches on a CT scan of a patient with a brain hemorrhage. Observe that the damaged region is located in the second slab from top to bottom.

In Figures 5(c) and (d), the results obtained with the volume partitioning and the slice grouping approaches applied to an histologic data model are shown. It is important to emphasize that these techniques have been applied without prior pre-processing. Time cost for computing the most structured view, and for volume partitioning and slice grouping are given in Table 1. Times are

Table 1. Time cost (seconds) for computing the most structured view, and for volume partitioning (P) and slice grouping (G) in 2, 4, and 8 slabs, respectively

Volume	Size	Best View	P(2)	P(4)	P(8)	G(8)	G(4)	G(2)
Human body	256 × 256 × 415	111.0	0.4	0.9	2.0	28.9	36.2	70.5
Hemorrhage	512 × 512 × 45	55.5	0.2	0.6	1.2	6.3	10.8	14.4
Histological data	587 × 342 × 499	722.6	2.3	5.8	12.6	214.0	324.7	525.0

given for a CPU Intel(R) Core(TM)2 Quad CPU Q6600 at 2.40GHz with 2 GB of memory. Several video sequences are available as supplementary material in `http://www.gametools.org/smartgraphics/`.

6 Conclusions

New partitioning techniques for volumetric data decomposition and visualization using exploded views have been introduced. These techniques use an information-theoretic two-step approach to automatically partition the model. First, the view with the highest structure is identified and, then, the model is divided along this view following two alternative similarity-based methods. The presented techniques provide us an efficient tool for volume data exploration without neither a priori knowledge nor pre-processing of the data. In our future work we will study how many slices should be presented and whether additional information about the similarity distances can help the user for an optimal understanding of the volume. Also, the trade-off between quality and cost for the different parameters involved in the determination of most structured views (distances between rays, subsequences of samples on those rays, sub-sampling of intensity values) will be investigated.

Acknowledgements. This work has been supported by TIN2007-68066-C04-01 and TIN2007-67982-C02 of the Ministry of Education and Science (Spanish Government), by the MedViz Initiative in Bergen (medviz.uib.no), and by the Austrian Science Fund (FWF) grant no. P18322. The visible human data set is courtesy of the Visible Human Project, National Library of Medicine, US.

References

1. Rheingans, P., Ebert, D.: Volume illustration: Nonphotorealistic rendering of volume models. IEEE Trans. on Visualization and Computer Graphics 7(3), 253–264 (2001)
2. Wang, S.W., Kaufman, A.E.: Volume sculpting. In: Proc. the Symposium on Interactive 3D Graphics, pp. 151–156 (1995)
3. Weiskopf, D., Engel, K., Ertl, T.: Interactive clipping techniques for texture-based volume visualization and volume shading. IEEE Trans. on Visualization and Computer Graphics 9(3), 298–312 (2003)

4. Konrad-Verse, O., Preim, B., Littmann, A.: Virtual resection with a deformable cutting plane. In: Proc. of Simulation und Visualisierung, pp. 203–214 (2004)
5. Viola, I., Kanitsar, A., Gröller, M.E.: Importance-driven feature enhancement in volume visualization. IEEE Trans. on Visualization and Computer Graphics 11(4), 408–418 (2005)
6. Bruckner, S., Grimm, S., Kanitsar, A., Gröller, M.E.: Illustrative Context-Preserving Exploration of Volume Data. IEEE Trans. on Visualization and Computer Graphics 12(6), 253–264 (2006)
7. McGuffin, M., Tancau, L., Balakrishnan, R.: Using deformations for browsing volumetric data. In: Proc. IEEE Visualization, pp. 401–408 (2003)
8. Vázquez, P.P., Feixas, M., Sbert, M., Heidrich, W.: Viewpoint selection using viewpoint entropy. In: Proc. Vision, Modeling, and Visualization 2001, pp. 273–280 (2001)
9. Takahashi, S., Fujishiro, I., Takeshima, Y., Nishita, T.: A feature-driven approach to locating optimal viewpoints for volume visualization. In: IEEE Visualization 2005, pp. 495–502 (2005)
10. Bordoloi, U.D., Shen, H.W.: Viewpoint evaluation for volume rendering. In: IEEE Visualization 2005, pp. 487–494 (2005)
11. Viola, I., Feixas, M., Sbert, M., Gröller, M.E.: Importance-driven focus of attention. IEEE Trans. on Visualization and Computer Graphics 12(5), 933–940 (2006)
12. Bruckner, S., Gröller, M.E.: Volumeshop: An interactive system for direct volume illustration. In: Proc. IEEE Visualization 2005, pp. 671–678 (2005)
13. Tishby, N., Pereira, F.C., Bialek, W.: The information bottleneck method. In: Proceedings of the 37th Annual Allerton Conference on Communication, Control and Computing, pp. 368–377 (1999)
14. Slonim, N., Tishby, N.: Agglomerative information bottleneck. In: NIPS, pp. 617–623 (1999)
15. Cover, T.M., Thomas, J.A.: Elements of Information Theory. Wiley Series in Telecommunications (1991)
16. Feldman, D., Crutchfield, J.: Structural information in two-dimensional patterns: Entropy convergence and excess entropy. Physical Review E 67 (2003)
17. Burbea, J., Rao, C.R.: On the convexity of some divergence measures based on entropy functions. IEEE Trans. on Information Theory 28(3), 489–495 (1982)
18. Viola, P.A.: Alignment by Maximization of Mutual Information. PhD thesis, Massachusetts Institute of Technology, Massachusetts (MA), USA (1995)
19. Maes, F., Collignon, A., Vandermeulen, D., Marchal, G., Suetens, P.: Multimodality image registration by maximization of mutual information. IEEE Trans. on Medical Imaging 16(2), 187–198 (1997)
20. Studholme, C.: Measures of 3D Medical Image Alignment. PhD thesis, University of London, London, UK (1997)

Hardware-Accelerated Illustrative Medical Surface Visualization with Extended Shading Maps

Christian Tietjen, Roland Pfisterer, Alexandra Baer, Rocco Gasteiger, and Bernhard Preim

Otto-von-Guericke University, Magdeburg, Germany
tietjen@isg.cs.uni-magdeburg.de

Abstract. In this paper, we introduce a new framework for the illustrative visualization of medical surface data. In most visualization frameworks, only light intensity is used to determine the surface shading. The analysis of medical textbooks reveals more complex shading approaches. The parameters of these approaches are mapped to different *Shading Maps*, which may be weighted and flexibly combined. We discuss the use of high-level attributes to simplify the specification. The resulting *Shading Map* is used as a lookup to determine the final intensity at a certain area. For this purpose, the rendering is accomplished on GPU by using OpenGL's Framebuffer Objects. This framework may be useful for interactive educational systems or for medical record printings.

1 Introduction

For illustrative visualizations, it is essential to convey the rich information on the object's shape and their relations to each other. Illustrators use shading to support shape perception, to guide the observer's attention and to achieve a balanced overall impression. We analyzed anatomical drawings to identify composition parameters and to adopt these to 3D surfaces. Parameters may describe local attributes of the objects surface or global attributes, like the relationship of different objects to each other.

The paper at hand discloses additional or alternative parameters for shading, regardless of the viewing direction, with the best idea of the scene. *Shading* in the further course describes the general use of brightness for displaying objects, as opposed to pure illumination shading. Every parameter is mapped onto one *Shading Map*, which may be weighted and combined. The resulting Shading Map is used as a lookup for the final intensity at a certain area. Therefore, a scene consisting of several medical surface objects may be rendered with stippling or hatching techniques, for example. It is also possible to control other properties like transparency, hue or saturation.

Since more freedom in parameterization leads to a greater demand of user interaction, we propose a simplification of the parameterization effort. Our framework is based on a two-level approach: at the higher level, the user may adjust

A. Butz et al. (Eds.): SG 2008, LNCS 5166, pp. 166–177, 2008.

four high-level attributes. At the elementary level, 20 parameter adjustments and weightings are available for fine-grained control.

This framework may be useful for interactive educational systems or for medical record printings. In most clinical offices still black-and-white printers dominate. Thus, an improved visualization for printings is helpful. Also, educational systems require motivational aspects in order to introduce the system or to explore the anatomical datasets. To render the scene in real-time, all Shading Maps are computed and combined on the GPU by using Framebuffer Objects.

2 Related Work

Bruckner et al. [1] present the concept of style transfer functions for illustrative volume rendering. Their image-based lighting model uses sphere maps to represent illustrative rendering styles. Tietjen et al. [2] introduce a combination of volume-, surface- and object-based line rendering. Svakhine et al. [3] build up a system to semi-automatically create visualizations of volume data in a variety of illustration motifs. The motif specification contains a compact set of core parameters. Ritter et al. [4] developed a pure black-and-white rendering of vascular structures to display shadow-like depth indicators and distance information for intra-operative use.

A couple of concepts exist for illustrative shading techniques. Akers et al. [5] developed a system to generate comprehensible images based on several photos. For non-photorealistic rendering of 3D objects, Hamel [6] suggests a lighting model consisting of different weighted illustrative shading techniques and simulates raking light by darkening surface areas with high curvature. Lee and Hao [7] optimize the overall lighting of a scene by using several local light sources. Shacked and Lischinski [8] propose a method to automatically determine the values of various lighting parameters by optimizing a perceptual quality metric. A similar approach is presented by Gumhold [9]. Both concepts are not designed for interactive exploration.

Yuan et al. [10] perform all non-photorealistic rendering operations in a geometry-image domain. In our approach, we need to compute the optimal shading per view, so our work is more related to Saito and Takahashi [11]. They make use of rendering buffers, so called G-buffers, to generate illustrative images including simple hatching techniques. Luft et al. [12] utilize the difference between the original depth buffer content and a low-pass filtered copy to enhance the perceptual quality of images. Rusinkiewicz et al. [13] investigate a non-photorealistic shading model based on adjusting the effective light position for different areas of the surface. Scheuermann and Hensley [14] present an efficient algorithm to compute image histograms in real-time entirely on the GPU. For the emulation of atmospheric depth, Ebert and Rheingans [15] suggest to interpolate non-linear between the original color and the background color value according to the depth value.

Illustrative, texture-based shading techniques are presented by [16,17]. Baer et al. propose a novel approach for frame-coherent and seamless scalable stippling.

Praun et al. introduce a real-time system for rendering of hatching strokes over arbitrary surfaces.

3 Shading Arrangement in Medical Textbooks

Numerous publications deal with the creation of meaningful illustrations by using graphical elements, like stipples or strokes. Less attention was payed to the shading arrangement of such black-and-white illustrations. Therefore, this section analyzes the arrangement of shading in medical textbooks.

3.1 Shape Modeling

In contrast to a photographer, the illustrator has the possibility to arrange the lighting proportions to achieve an optimal perceptibility of the structure's shape and surface. For a detailed analysis of these methods, we refer to Hodges [18].

Conventional and Reflected Lighting: A light source from the top left is used corresponding to the perception of the daylight. In reality, *reflected light* ensures that shadow territories appear not completely black. The *reflected light* is used as scatter lighting to make structures apparent that are not directly lighted.

Plateau Lighting: *Plateau lighting* is suitable to emphasize the silhouette and for figure-ground-separation. This results in an enhanced shape perception. For roundish structures, the center appears brighter and border areas are darker.

Backlighting: In contrast to the *plateau lighting* the *backlighting* separates an object from a dark background. The *backlighting* has a lower effect than the *plateau lighting* and may be used at the same time.

Raking Light: Details are optimally recognizable when the light is approximately tangential to the surface, although the *raking light* does not affect the overall impression of a realistic main light source.

For the illustration of the object's surface, additional techniques are used, which are not related to the incidence of light. *Ridges and valleys* may be especially highlighted to improve the conventional lighting methods.

3.2 Depth Cueing and Contrast Enhancement

Because a 2D illustration has no third dimension, illustrators are using optical hints to visualize the depth of the scene more clearly. In nature, the scattering of light and fog are the reason that farther objects appear brighter and with low contrast. The *atmospheric perspective* gives a rough idea of depth and clarifies the whole scene, because fore- and background are optically separated.

Furthermore, the distance between overlapping objects may be emphasized by mapping a *shadow* onto the rear object. This *shadow* seems to be caused by a light source from the direction of the observer. The shadow is independent of the *main light* source and is only applied locally. Thus, it differs from a shadow in reality. Additionally, the front object is brightened at the boundaries to aid the contrast to the rear object, similar to the *backlighting*.

In addition, a *high contrast range* makes illustrations appear more dynamic and increases the amount of perceived information at the same time. Therefore, the whole contrast range should be utilized.

4 Emulation of Shading Arrangements

Techniques like stippling or hatching are based on the placement of black stipples or strokes to create an impression of gray values. Thus, the actual gray value of one resulting pixel is black or white, but for every pixel a target gray value must be determined. Typically, the stippling or hatching renderer computes the final gray value based on a local lighting model. In the following, a system for a flexible computation of an illustrative shading is described.

4.1 Requirement Analysis

The goal of medical illustrations is to depict the human anatomy expressively. The emphasis is therefore placed on the illustration of shape and spatial relationships. Since the computer generated illustrative shading for 3D scenes is an extension of the medical textbook illustrations, the already discussed techniques are well suited.

To adopt the computer generated shading to different geometric models, it is necessary that the different shading arrangements are provided as parameters in an appropriate user interface. Since the parameters may have different impact on the visualization, a differentiated examination of their involvement for the final rendering is necessary.

This results in a high-dimensional parameter space. Therefore, a guided interaction is essential which provides a good trade-off between illustrative freedom and simplicity of use.

4.2 Realization of the Extended Shading

The algorithms for computing the single parameters pose different computational requirements. The whole pipeline is completely passed to the GPU to achieve interactive frame rates. The concept of Framebuffer Objects (FBO) is used [19]. Every parameter is mapped on one FBO. To commit additional static values to the GPU, e. g. curvature, user vertex attributes provided by OpenGL are used.

Lighting is computed per fragment and therefore for every point of the surface. However, histogram operations have to be computed for the whole image. Hence, the local brightness of a surface is dependent on the surrounding. It is not possible to render the brightness, as customary, in a normal fragment shader. Therefore, we suggest to compute the brightness distribution in a separate process and make it available in the final rendering process.

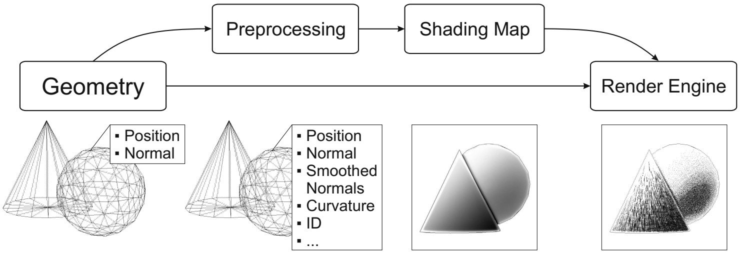


Fig. 1. View-independent values are mapped onto the surface and rendered into shading maps. The combined shading maps are used for brightness for the final rendering.

Some parameters are computed more efficiently in object space, e. g., lighting and curvature, others in image space (like histogram operations). For the combination of both spaces it is necessary to transfer them into a uniform space. Therefore, the image space is appropriate, because the needed object parameters are computable in image space, according to the G-buffers [11].

All static values are computed once in object space, to save computation for the single rendered frames. The combination of all parameters yields to the so-called extended shading map: a view of the scene with current camera position and screen resolution, indicating the brightness for every single pixel. The final renderer reads the brightness value per pixel and produces the actual rendering using the desired technique, like stippling, hatching or color saturation (Fig. 1).

According to the G-buffers, for each parameter an individual parameter buffer is generated and stored in a FBO. Some of these buffers are generated directly by reading the hardware-buffers (e. g., z-buffer), others are computed after a further image processing.

4.3 Buffer Combination

The sequence used in the combination of all parameter buffers significantly influences the resulting image. In the following equations parameter buffer P and shading map M are regarded as 2D arrays with corresponding weights $w = [0, 1] \in \mathbb{R}$. The addition of two parameter buffers $P = P_i + P_j$ is performed element-wise:

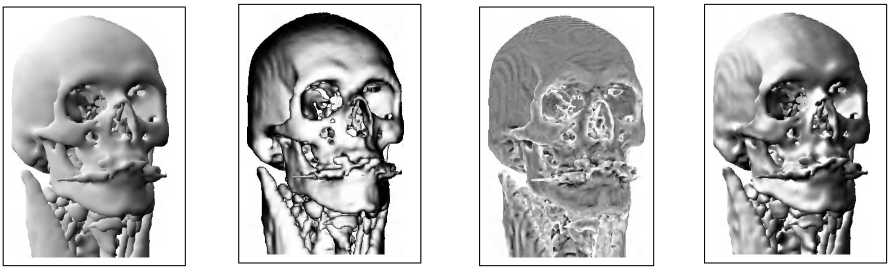
$$\forall x, y : P(x, y) = P_i(x, y) + P_j(x, y) \tag{1}$$

The product $P = w \cdot P_i$ is accordingly:

$$\forall x, y : P(x, y) = w \cdot P_i(x, y) \tag{2}$$

The first parameters, that are *conventional*, *reflected*, *plateau* and *raking light*, as well as *surface shape* and *curvature*, represent the distribution of brightness. According to [5,6], the parameter buffers are added up to a weighted sum:

$$M = \frac{1}{\sum w_i} \cdot \sum w_i \cdot P_i \tag{3}$$



(a) Main light (b) Plateau light (c) Curvature (d) Combination

Fig. 2. Combination of the surface representation parameter: (d) is the weighted sum of (a), (b) and (c), and an additional histogram equalization

The resulting shading map M is normalized and subsequently modified by a *histogram equalization* to enhance the contrast (Fig. 2(d)). The equalization has to be performed at this stage, because all of the following parameters only change the local brightness. If they are added before the equalization, the local changes affect the whole scene. For example, the *atmospheric perspective* would also darken the foreground.

The following parameter buffers need individual considerations due to the different characteristics. Therefore, the sequence of modifications has to be regarded. In the following, the result M' accords to M in the subsequent equation.

Although *backlighting* and *depth shadow* are lighting techniques, they are added separately, because their impact should be limited to the object's boundaries. An inclusion of these parameters into the weighted sum would result in a lower contrast, as explained above. Instead, the *backlighting* is added to the assembled shading map and the result is clamped to $[0, 1]$:

$$M' = \underset{[0,1]}{clamp}\,(M + w \cdot P) \tag{4}$$

The depth shadow buffer should only influence pixels in shadow regions. According to [12], the buffer values lie in the range $[-1, 0]$. With the following equation, the brightness is shifted to 0 in shadow regions and other regions are left unattended:

$$M' = (1 + w \cdot P) \cdot M \tag{5}$$

The *atmospheric perspective* is realized by contrast reduction according to [15]. In addition, the *background intensity* $I_b \in [0, 1]$ is freely adjustable:

$$M' = (1 - w_{ap} \cdot P_{ap}) \cdot M + w_{ap} \cdot P_{ap} \cdot I_b \tag{6}$$

The integration of *feature lines* is carried out at last, because the lines should not be effected by any other parameter. Additionally, the thickness of the *feature lines* may be varied by a weight. Due to the functionality of edge filters, the buffer has to be inverted after weighting. The feature lines are added by the minimum operation to ensure their display on bright areas and to avoid the brightening of dark areas:

$$M' = \min\left(1 - w \cdot P, M\right) \tag{7}$$

It is also possible to convey the *atmospheric perspective* by varying the *feature line* thickness. The size of the edge filter is chosen according to the value of the *atmospheric perspective.*

5 Parameter Adjustment and Simplification

All specified parameters will affect the visualization. Again, the impact of a parameter is controlled by a weight and possibly adjustments. Thus, to obtain the final visualization, 20 adjustments and weights have to be tuned (Fig. 3). Therefore, default values and the summary of parameters to high-level attributes have to be determined.

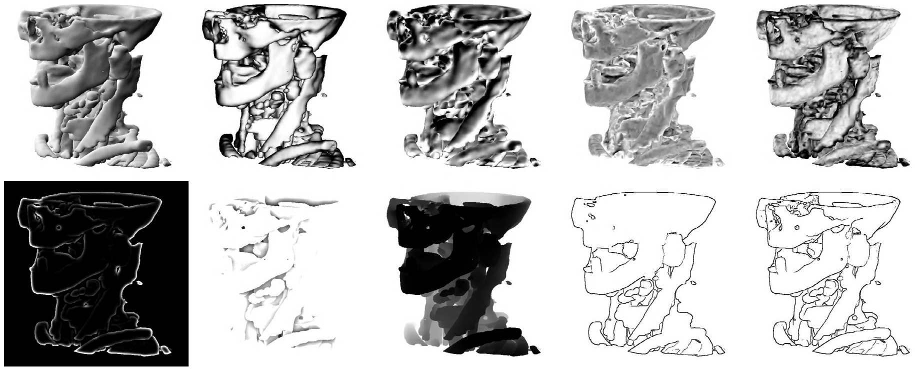


Fig. 3. From top left to bottom right: conventional, reflected, and plateau lighting, raking light, surface shape, curvature, backlighting (black background), depth shadow (increased by 1), atmospheric perspective, silhouettes (inverted), feature lines (inverted)

5.1 Parameters for the Shading Arrangement

So far, we introduced some parameters for illustrative shading. In the following, for every parameter a realization is described and the adjustment is discussed. To support a flexible parameterization, no combined light models are used, as recommended by [7]. Instead, the single lighting techniques are applied separately to achieve an efficient parameter controlling.

The *conventional and reflected lighting* is achieved by a modified diffuse lighting according to [20], where no separate parameter for the *reflected light* is necessary. The direction of the *main light* is selectable. Additionally, the usage of *normal smoothing* is possible to get a softer shading. The *plateau lighting* has no adjustments and corresponds to diffuse lighting from the viewing direction.

Local lighting according to [13] emulates *raking light*. The direction is parallel to the *main light*. The parameters are *toon shading* and *detail frequency*.

The *surface shape* is emphasized by ridges and valleys. *Curvature* is visualized according to [6] by darkening surface areas with high curvature. Both parameters need no adjustment.

Backlighting is extracted from the unsharping mask of the depth buffer according to [12], but only positive values are considered. The size of the kernel is variable. Like *backlighting*, the *depth shadow* is also extracted from the unsharping mask of the depth buffer, but only negative values are considered. The *atmospheric perspective* is computed by the depth buffer according to [15]. Adjustments relate to minimal and maximal depth, slope exponent as well as background intensity. *Silhouette and feature lines* are obtained by edge filtering of object and depth buffer according to [11]. The thickness of the lines is defined by filter type and dimension. *Histogram equalization* is chosen for contrast enhancement [14]. The equalization permits a frame-coherent rendering, in contrast to simple histogram stretching, and needs no adjustment.

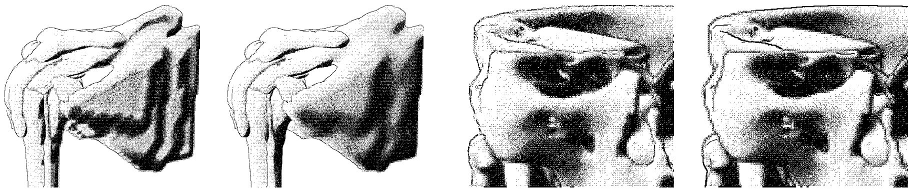
5.2 Default Values

The adjustment may be improved by providing adequate default values for the parameters, which the user will likely accept in most cases.

Detail Frequency: Since we consider a parameterization for medical surfaces, the resolution and quality of the objects is an important aspect. Strong artifacts should result in a low *detail frequency* of the *raking light* and an increased *smoothing* (Fig. 4(a)). The *detail frequency* cannot be abridged and must always be adjusted.

Weighting of the Feature Lines: *Feature lines* should not be too prominent to keep the illustrative character and to fit better into the overall rendering (Fig. 4(b)). A favorable value is 80% of the full intensity.

Direction of the Conventional Main Light: The *main light* shines from the top left to the lower right [18]. Because the main light vector is perpendicular to the viewing vector, the modified diffuse lighting is optimally used.



(a) Detail frequency

(b) Feature lines

Fig. 4. (a) A shoulder dataset with strong artifacts. Smoothing removes artifacts, but features also disappear. (a) Slightly attenuated feature lines fit better into the overall rendering.

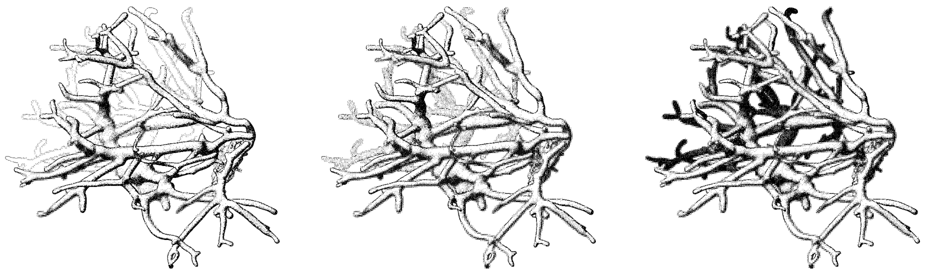


Fig. 5. A liver vessel tree with atmospheric perspective and different background intensities. The best attenuation is achieved on the left.

Weighting of the Raking Light: To maintain the impression of the *main light*, the *raking light* weight should be in a 1 : 2 ratio to *main and plateau light.* When this ratio is maintained, the *toon shading* adjustment alone is sufficient, thus, the parameterization is further simplified.

Background Intensity: According to medical textbooks, the background intensity of the *atmospheric perspective* is white, otherwise the background gains more attention (Fig. 5).

5.3 High-Level Attributes

Individual parameters may be combined to high-level attributes. For example, different parameters influence the smoothness of the surface and the border enhancement. By a combination of these parameters, the user's effort is reduced significantly to only one weight. The *detail frequency* is also kept as a high-level attribute, since it cannot be abridged and must always be adjusted.

Border Enhancement: Besides the *feature line* thickness, the ratio between *main and plateau light* affects the intensity of the object's border. The ratio may be combined with feature line thickness to one high-level attribute (Fig. 6).

Detail Amplitude: Details are emphasized by the toon shading factor of the *raking light* as well as the *curvature weight* and extenuated by *normal smoothing* of the *main light.* These parameters are combined to a detail amplitude (Fig. 7). The extremes correspond to high details and additional smoothing respectively. The medium setting is according to a normal lighting.

Spatial Effect: *Back light*, *depth shadow* and *atmospheric perspective* contribute to the spatial effect of a scene. Thus, their weights may be combined to one high-level attribute.

5.4 Effectiveness of the Manual Parameterization

The presented approaches aim at a more simple parameterization for the user. Therefore, it is essential to verify whether a simple parameterization could be actually achieved. It is difficult to evaluate how long a user needs to achieve the

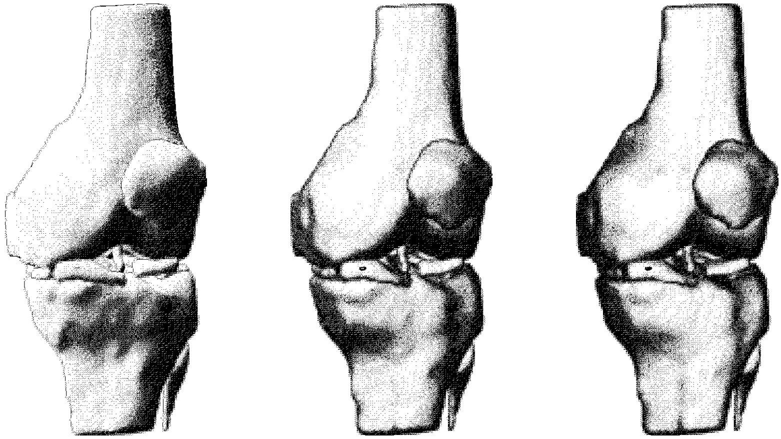


Fig. 6. Border enhancement of a knee dataset from left to right: main light decreases, feature line thickness and plateau light increases

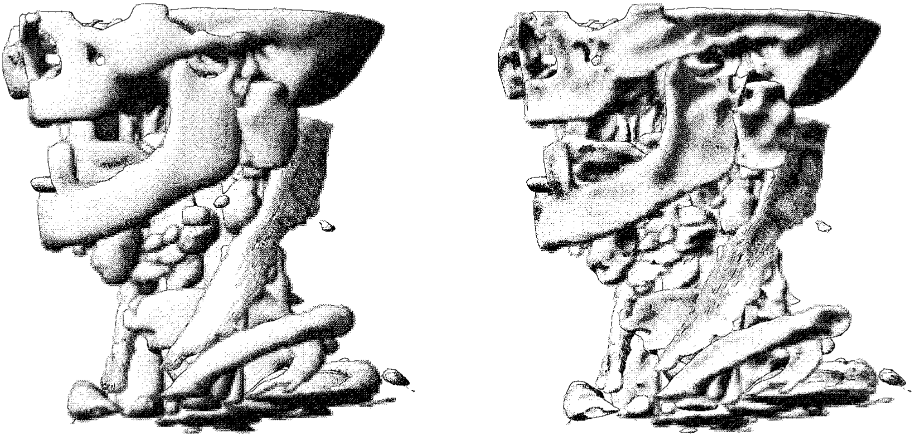


Fig. 7. Detail amplitude: on the left, only main light and smoothed normals are used. Rightwards, smoothing decreases, toon shading and curvature weight increases. For the muscles, a hatching technique was applied.

desired shading, since the appreciated quality depends not only on the specific application example, but also on the personal aesthetic sense. Instead, we investigated the effectiveness of the parameterization, i. e., how goal-oriented the user may change the parameterization.

Since every modification at the shading is immediately visible in real-time, the impact of every single parameter is directly observable. In principle, the user may give a trial on the impact of all parameters. Without any simplification 20 parameter adjustments and weights are available and the contributions of the parameters partially reinforce or interfere. Thus, the application of default values and high-level attributes is highly recommended.

By using the high-level attributes, the parameterization is reduced to four settings. The procedure is additionally eased by the following strategy:

1. Border Enhancement
2. Detail Amplitude
3. Detail Frequency
4. Spatial Effect

The impact of these attributes is almost independent from each other. Due to the descriptive change of the comprehensive setting, even a novel user should be able to achieve a final setup very fast.

6 Conclusions and Future Work

We presented a new approach for an illustrative shading framework. For an efficient and flexible parameterization, we introduced single parameters and high-level attributes, which incorporate a wide variety of illustrative styles. The framework is completely hardware-accelerated by the GPU using framebuffer objects. Thus, real-time rendering is achieved even for very complex geometric models.

We tested our framework with all the scenes presented in this paper with different viewport sizes (Tab. 1). The number of triangles ranges from $25K$ to $225K$. The testing environment was a workstation with $3.2GHz$ CPU, $1GB$ RAM and a NVidia GeForce 7900GS graphic card. The variation of the parameters has no impact on the frame rates. The frame rates are measured for the shading map generation process only. Stippling reduces the frame rate by 5%.

Table 1. Frames per second for different scenes and a viewport size of 512^2. Extended shading maps are compared to conventional shading.

Model	Fig. 2	Fig. 4(a)	Fig. 5	Fig. 6	Fig. 7
Triangles	147480	39863	225080	25224	94399
Ext. shading	10	15	7.5	20	14
Conv. shading	30	60	15	60	45

In the future, we intend to consider an *importance* parameter to further refine our visualizations. To simplify the parameterization, other interaction facilities will be explored, e. g., to represent a high-level attribute in a graphical editor. We want to overcome the limitation of opaque surfaces. So far, first tests brought good results by using depth peeling. A step towards an automatic parameterization would be a similar approach to [8]. Also, methods for other shading improvements used in illustrations, like composition, balance and harmony, could improve the quality of the presented images.

Acknowledgment. We thank MeVis Research for providing advanced MeVisLab features. This work was supported by the Deutsche Forschungsgemeinschaft (DFG) (Priority Programme 1124, PR 660/3-2).

References

1. Bruckner, S., Gröller, M.E.: Style Transfer Functions for Illustrative Volume Rendering. Computer Graphics Forum 26(3), 715–724 (2007)
2. Tietjen, C., Isenberg, T., Preim, B.: Combining Silhouettes, Shading, and Volume Rendering for Surgery Education and Planning. In: EuroVis., pp. 303–310, 335 (2005)
3. Svakhine, N., Ebert, D.S., Stredney, D.: Illustration motifs for effective medical volume illustration. IEEE Computer Graphics & Applications 25(3), 31–39 (2005)
4. Ritter, F., Hansen, C., Dicken, V., Konrad, O., Preim, B., Peitgen, H.O.: Real-Time Illustration of Vascular Structures. IEEE TVCG 12(5), 877–884 (2006)
5. Akers, D., Losasso, F., Klingner, J., Agrawala, M., Rick, J., Hanrahan, P.: Conveying Shape and Features with Image-Based Relighting. In: IEEE Visualization, pp. 349–354 (2003)
6. Hamel, J.: A New Lighting Model for Computer Generated Line Drawings. PhD thesis, University of Magdeburg (2000)
7. Lee, C.H., Hao, X.: Geometry-Dependent Lighting. IEEE TVCG 12(2), 197–207 (2006)
8. Shacked, R., Lischinski, D.: Automatic Lighting Design Using a Perceptual Quality Metric. Computer Graphics Forum 20(3), 215–227 (2001)
9. Gumhold, S.: Maximum Entropy Light Source Placement. In: IEEE Visualization, pp. 275–282 (2002)
10. Yuan, X., Nguyen, M.X., Zhang, N., Chen, B.: Stippling and Silhouettes Rendering in Geometry-Image Space. In: Eurographics Symposium on Rendering, Eurographics Association, pp. 193–200 (2005)
11. Saito, T., Takahashi, T.: Comprehensible Rendering of 3-D Shapes. Computer Graphics 24(4), 197–206 (1990)
12. Luft, T., Colditz, C., Deussen, O.: Image Enhancement by Unsharp Masking the Depth Buffer. ACM Transactions on Graphics 25(3), 1206–1213 (2006)
13. Rusinkiewicz, S., Burns, M., DeCarlo, D.: Exaggerated Shading for Depicting Shape and Detail. ACM Transactions on Graphics 25(3), 1199–1205 (2006)
14. Scheuermann, T., Hensley, J.: Efficient Histogram Generation Using Scattering on GPUs. In: Proc. of the 2007 Symposium on Interactive 3D Graphics, pp. 33–37 (2007)
15. Ebert, D.S., Rheingans, P.: Volume Illustration: Non-Photorealistic Rendering of Volume Models. In: IEEE Visualization, pp. 195–202 (2000)
16. Baer, A., Tietjen, C., Bade, R., Preim, B.: Hardware-Accelerated Stippling of Surfaces Derived from Medical Volume Data. In: EuroVis., pp. 235–242 (2007)
17. Praun, E., Hoppe, H., Webb, M., Finkelstein, A.: Real-Time Hatching. In: SIGGRAPH, pp. 579–584 (2001)
18. Hodges, E.R.S.: The Guild Handbook of Scientific Illustration. Van Nostrand Reinhold (1989)
19. Juliano, J., Sandmel, J.: OpenGL Framebuffer Object Extension (December 2007)
20. Gooch, A., Gooch, B., Shirley, P., Cohen, E.: A Non-Photorealistic Lighting Model for Automatic Technical Illustration. In: SIGGRAPH, pp. 447–452 (1998)

Smart Lenses

Conrad Thiede, Georg Fuchs, and Heidrun Schumann

Institute for Computer Science
University of Rostock
{firstname.lastname}@uni-rostock.de

Abstract. Focus + context techniques are widely used for the efficient visualization of large data sets. However, the corresponding adaptation of the representation to the task at hand is not trivial, requiring a suitable model of the visualization goal. One type of focus + context technique is the use of lenses, interactive tools that modify the visualization in a locally confined region and can be 'stacked' to create complex filters. In this paper we propose a new approach to intergrate *smart lenses* into the visualization process based on Chi's Data State Reference Model.This allows us to automatically adapt specific aspects of the visualization based on relevant influence factors, without complex task modeling.

1 Motivation

The size of today's typical data sets is increasing steadily, requiring new strategies in exploring the data to gain insight in an efficient manner. For this purpose information visualization is a well-known approach for its ability to support the analysis of huge data sets. Yet the limited screen size does not allow to show the entire data set in full detail as this would inevitably cause visual cluttering [1]. Consequently, we must show the data with different granularity based on its importance with respect to a given user's goal.

A common approach to this end is the concept of *focus + context*. Here, a region of interest (RoI) is defined as focus that encloses the subset of data elements most relevant to the user, shown in full detail. The context provides an overview of the remaining data with reduced fidelity, and serves to maintain user orientation when navigating the data set. However, deciding what data is relevant is by no means a trivial problem. The automatic RoI specification requires a suitable model of the user's task at hand [2]. For this reason, many approaches rely on the interactive selection of the RoI by the user instead, delegating to her the decision what data is relevant.

One representative of such focus + context techniques are the so-called *lenses*. These are filters that are placed interactively by the user and alter the visualization in a locally confined region [3]. Multiple lenses can also be 'stacked' to combine filter effects [3,4]. Moreover, applying a filter only to a contained RoI is on principle more efficient than applying it globally.

In this paper, we present a concept for the definition of *smart lenses* based on the the Data State Reference Model (DSRM) by Chi [5], understanding lenses as

A. Butz et al. (Eds.): SG 2008, LNCS 5166, pp. 178–189, 2008.

individual operators within this framework. Smart in this context means that no holistic task model is required. Instead, single lenses can be employed to modify specific aspects of the visualization within the current RoI. We also provide a systematization of the influence factors that guide the selection and adequate parametrization of those lenses together with some motivating examples.

The paper is structured as follows. In Sect.2 the DSRM is briefly introduced and examples for lens operators on its different stages are given. Section 3 discusses our approach for lens definitions in detail, while Sect. 4 gives examples for their application. Sect. 5 gives a summary and some concluding remarks.

2 The Data-State-Reference Model

Our approach is based on Chi's Data State Reference Model (DSRM) [5]. It describes the visualization process as a pipeline of four *stages* (data states), which are sets of data elements with attributes, and *operators* on the data (Fig. 1). The four stages are, in order of traversal:

1. **Raw data**, i.e. the input data elements with attribute values.
2. **Analytical abstractions**, obtained from raw data values e.g. by calculating statistical moments. Data on this stage is sometimes referred to as meta data.
3. **Visual abstractions**, i.e. high-level visualization concepts such as Scatterplots or 3D surface representations (e.g. triangle meshes).
4. **Image data** that is the result of the rendering process, i.e. pixel data.

Data is transformed and propagated through the pipeline by operators of two different types, *stage operators* and *transformation operators* [5]. Stage operators work within a single data stage, while transformation operators transfer data from one state to another (cf. Fig. 1). The DSRM further distinguishes between three different types of transformation operators based on the pipeline stages they connect:

- **Filtering** operators transform raw data values into analytical abstractions. This for example includes interpolation of missing values, removal of erroneous readings, and the calculation of mean and extreme values etc.

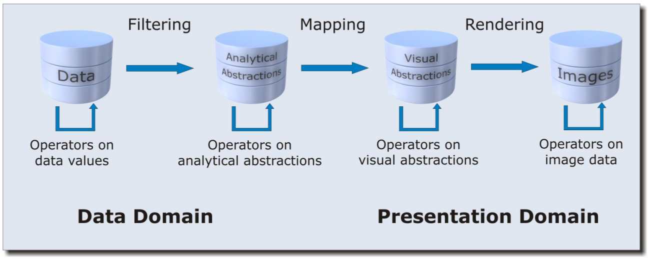


Fig. 1. Data State Reference Model [5] forming the framework for lens integration

- **Mapping** operators take analytical abstractions as input and map these to visual abstractions, e.g. by means of color mapping.
- **Rendering** operators process the high-level visual abstractions in order to obtain the visual representation.

In this context, we can also understand lenses as operators in the DSRM, albeit spatially confined to the current RoI [4]. Furthermore, the data states can be divided into two groups [5]. The first two states define the (abstract) data domain, whereas the last two describe data in the presentation domain (Fig. 1). This corresponds to the distinction between semantic and graphical lenses [6]. The versatility of this operator-based concept for lens definitions is illustrated by the following examples.

- Lenses as *stage operators* on *data values* can control the ratio of visualized values in the lens region to avoid clutter (sampling, e.g. [1,7]), remove errors or interpolate new data.
- Lenses as operators on *analytical abstractions* can be used to sample the corresponding data elements on this stage, as well as to show additional informations about the data, e.g. characteristics of the cluster(s) in the RoI.
- Lenses as operators on *visual abstractions* can for example change the Level-of-detail (LoD) of a given surface mesh [8,9] or generalize (simplify, aggregate) shapes in the lens region [10].
- Lenses as operators on *image data* realize standard per-pixel image modifications like hue or contrast adjustment, which e.g. can be used to overcome color vision deficiencies, or pixel-based distortions [11].
- Lenses as *filter operators* offer the possibility to calculate additional data characteristics on demand, like node metrics in a large graph [12], and vice versa, to filter out unimportant information like crossing edges to reduce clutter [13].
- Lens as *mapping operators* modify the mapping process and thereby decide about the type of visual representation. This can be used to visualize additional or different other aspects of the data in the RoI than in the remaining image, e.g. [3,8,14]
- Lenses as *rendering operators* realize graphical lenses, for example distortion lenses like Rase's Cartographic Lens [15,16].

Furthermore there are lens functions that consist of more than one operator on multiple pipeline stages, if for example the modification of the visual representation (3rd stage) is done based on semantic information from the data domain (1st or 2nd stages). A representative for this kind of functions is the Semantic Depth-of-Field approach by Hauser et al. [17].

As the examples show, lenses as operators can be applied on every stage of the model in a useful way. In doing so, there are basically two ways to integrate a lens into the visualization process [1]:

- Two separate pipelines are used, one for creating the underlying visualization and one for the lens region.
- The entire image including the lens region is produced using a single pipeline. The lens function alters data on the different stages directly (cf. [1]).

Although with the first approach it is easier to modularize it can not be used for stacking lenses, as their pipelines process the data independently unless elaborate interprocess communication is used. Therefore, we opted to implement a single visualization pipeline that is modified directly by all active lenses [4], with each lens defined as a self-contained plug-in using a declarative script (cf. Fig. 2).

Our goal is to extend our existing approach from [4] by further automating aspects of the lens that previously were left to the user. This will make lenses smart in that, based to the user's goal, only specific aspects of the visualization are adapted within the current RoI.

3 Specification of Smart Lenses

3.1 General Lens Definition

A lens is defined by three parameters *position*, *shape* and the *lens function*.

The *position* of a lens specifies the location of the lens in the visual representation. This is either a 2D point on the virtual canvas in case of a 2D visualization, or 3D coordinates in virtual viewspace.

The *shape* defines the boundary of the RoI enclosed by this lens. Usually, the shape is defined through simple geometries such as rectangles, circles and cubes, spheres for the 2D and 3D cases, respectively. However, more complex shapes are also possible, e.g. polygons with holes. In conjunction with the lens position, the shape therefore uniquely defines the *region* that is affected by the lens function *in view space* (i.e., an area or volume). As a corollary to this, the lens' region changes every time either its position or shape change. Moreover, due to the functional dependencies imposed by the DSRM operators, the shape always partitions the data elements *of each pipeline stage* into two disjoint subsets: the inside of the lens region contains all data elements affected by the lens function, and the second set contains everything else. A lens shape defined in the view space must therefore be mapped into the lower stages if influenced by a corresponding operator. This is a major issue and will be revisited in Section 3.1.

Lastly, the *lens function* can add, remove and modify data elements on each stage of the DSRM just like any other operator. Hence the lens function specifies how the elements within the lens region are processed.

Usually the user specifies these lens parameters interactively with regard to her interests. Our goal is an automatic parameterization to the extend possible. This means a *smart lens* should be adapted automatically with regard to certain aspects of the task at hand. To this end, we first have to identify relevant aspects that may influence lens parameters. Then, meaningful strategies for the automatic parametrization based on those aspects need to be derived.

3.2 Automatic Adaptation of Lens Parameters

The relevant aspects with respect to an automatic lens parametrization can be broken down into three broad categories:

- **User-related aspects:** In general these aspects are specified by means of a user profile that contains information like the overall proficiency or more fine-grained preferences of the user. For example, the user might prefer a circular rather a rectangular lens shape for a given application.
- **Data-related aspects:** Here the lens is adapted with regard to specific data characteristics, e.g. including values similar to a reference value, or child elements of a selection in a hierarchy, in order to set both position and shape of the lens in a way that interesting elements are contained by the lens region. For static data sets as used in our examples, this can be achieved using appropriate filter conditions[1].
- **Task-related aspects:** These aspects reflect the user goals in the context of the tasks at hand. For localization tasks this can mean the automatic placements of lenses over conceivable regions of interest. For identification or comparison tasks, the lens function could for example automatically apply a locally optimized or a global color scale, respectively [18].

The examples above show that different aspects may influence the lens parameters, and that an automatic adaptation can support the user in exploring her data and solving the tasks at hand.

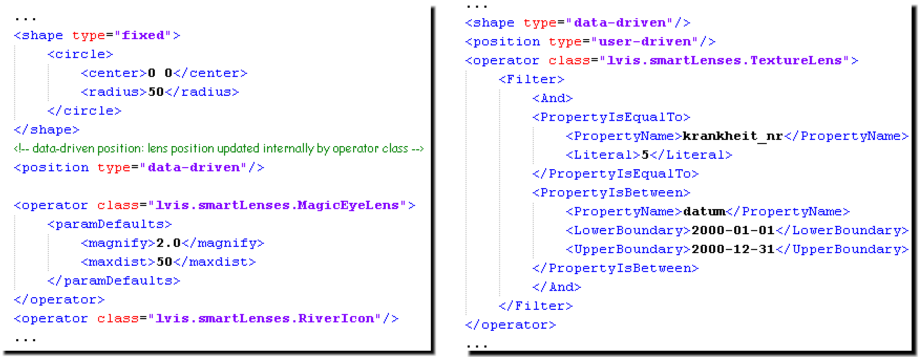
Now the question is how this automatic adaptation should be realized. For this purpose let us have a look at the automatic adjustment of user interfaces in the context of multiple user interfaces for different devices (see [19]). Here, the adaptation process is done based on different models. (i) user model includes the user-driven aspects, (ii) a task model specifies a hierarchy of compound tasks with subtasks and their (temporal, causal) relationships, describing the structure of the task to be solved, and (iii) a dialog model, often derived from the task model, is used to organize the UI in functional groups and to layout GUI elements in a platform-independent way [20].

Instead of applying such extensive model descriptions we propose a simpler strategy: to describe the aspects relevant for the parametrization of smart lenses within the declaration scripts for the corresponding plug-ins. Currently, we use one such script per lens and user that we extend to contain the following additional information (cf. Fig. 2):

- The preferred lens shape,
- the Region of Interest in the data domain, and
- the required lens functions.

The *preferred lens shape* was chosen to include at least one user-driven aspect. The *RoI specification* was chosen because it is important to consider data- and task driven aspects as well. A RoI in the data domain includes data elements from specific attribute value ranges of interest. This can be expressed by means of suitable filter conditions. For our prototype, we use a notation inspired by the OGC Filter Specification [21]. Its XML syntax allows to express complex nested conditions including spatial, arithmetic and logic constraints (cf. Fig. 2, right). A

[1] For dynamically changing data sets, a far more complex approach would be required.



```
...
<shape type="fixed">
    <circle>
        <center>0 0</center>
        <radius>50</radius>
    </circle>
</shape>
<!-- data-driven position: lens position updated internally by operator class -->
<position type="data-driven"/>

<operator class="lvis.smartLenses.MagicEyeLens">
    <paramDefaults>
        <magnify>2.0</magnify>
        <maxdist>50</maxdist>
    </paramDefaults>
</operator>
<operator class="lvis.smartLenses.RiverIcon"/>
...
```

```
...
<shape type="data-driven"/>
<position type="user-driven"/>
<operator class="lvis.smartLenses.TextureLens">
    <Filter>
        <And>
        <PropertyIsEqualTo>
            <PropertyName>krankheit_nr</PropertyName>
            <Literal>5</Literal>
        </PropertyIsEqualTo>
        <PropertyIsBetween>
            <PropertyName>datum</PropertyName>
            <LowerBoundary>2000-01-01</LowerBoundary>
            <UpperBoundary>2000-12-31</UpperBoundary>
        </PropertyIsBetween>
        </And>
    </Filter>
</operator>
...
```

Fig. 2. Examples of lens definition scripts. On the left is the definition of the lens from Fig. 4(left). The script on the right defines the Texture lens from Fig. 5, including a composite filter to specifiy a RoI in data domain.

task-driven specification of RoI also makes sense, e.g. focusing on specific spatial regions. To maintain the highest degree of flexibility we allow the specification of RoI at every stage of the data state reference model. The *lens function* can be selected from a list of predefined functions (i.e. available as plug-in classes, cf. Fig. 2). These functions can be seen as specific operators at different stages of the data state reference model.

Our approach can be summarized as follows: First the script has to be specified, from which the corresponding operators required in the DSRM for the smart lenses are generated. The script therefore abstracts from the actual implementation of the visualization process, and can be reused or modified for other data sets or other visualization settings. Conceptually, such a script can be auto-generated after relevant aspects have been queried from the user, e.g. using a wizzard-style dialog. Subsequently, the visual output is generated by a processing pipeline realizing the DSRM.

As already mentioned in Sect. 3.1, a major point in the context of operator specification is the definition and mapping of the RoI on the different stages. We will discuss this problem in more detail in the next subsection.

3.3 Description of the Lens Region on Every Stage of the DSRM

The script defines lens functions as operators within the DSRM. Thus, we need a valid description of the lens region at every stage of the DSRM. To achieve this goal we use the concept of *half spaces*, which is flexible enough to describe different shapes at different stages of the DSRM. Generally, a half-space is that portion of an n-dimensional space obtained by removing that part lying on one side of an $(n-1)$-dimensional hyperplane. Therefore, an arbitrary region in the n-dimensional (data) space can be defined as the intersection of appropriate half spaces. In the simplest case, this means to partition the data space along each relevant attribute axis twice, at the minimum and the maximum value of interest, respectively.

The challenge is to map such region definitions from one stage onto the other and vice versa. In particular, we have to consider the following two cases:

- The lens region has been specified at the first stage (raw data): This is the simplest case. We use the pipeline to map the lens region onto the following stages.
- The lens region specified at a later stage of the pipeline, but the specified lens function requires operators on previous stages: In this case we additionally need an inverse mechanism, or back-projection, to collect the data elements associated with this region on the earlier stages.

There are two approaches to the inverse mapping required for the second case:

- If for each lens operator an inverse operator exists, those inverse operators can be used to calculate the lens region on earlier stages of the DSRM. Usually, however, such inverse operators are not feasible at all, and in some other cases it is not an easy task to define them.
- Another idea is to use a multi-pass approach to specify the lens region on every stage. During the first pipeline pass, for the second and subsequent stages lookup tables are generated. These contain, for each data element on a given stage, the associated input elements for the transformation operators from the previous stage. During the second pass, data elements contributing to the lens regions on the different stages are processed according to the lens function. Besides requiring two passes, a major disadvantage of this approach is the potential memory requirements for the lookup tables, especially for large data sets.

Although the first alternative is faster and generally has a smaller memory footprint, we use the second approach. We can not assume that all inverse operators are given, and we want to assure getting a valid result in any case.

Up to now we have only considered the definition and mapping of a single lens region. In fact, several lens regions at different stages can be defined by scripts. If these lens regions overlap, it will be necessary to detect and solve potential conflicts.

3.4 Combination of Smart Lenses

Combining two (or more) *overlapping* lenses requires two steps. First, the lens regions have to be merged. Since we define lens regions as an intersection of half spaces, their unification is a trivial task.

The second step is the combined execution of the DSRM operators constituting both lens functions of the overlapping regions. This requires an additional effort. There are different approaches imaginable:

- Precedence: Only one lens function is applied to the overlapping region. Which function takes precedence can be determined by the order of specification or priority values from the script.

- Fusion of the lens functions: In this case a new lens function for the overlapping part is specified based on the operators constituting the original lens function. This requires a suitable and consistent function description, e.g. with predicate logic.
- Combination of function outputs: Both lens functions are independently applied to the overlapping region, the output element sets are merged (cf. [1]).

Because a description of the lens function using predicate logic is not always available we deepen the more general approach of combining lens function results.

Furthermore, if we combine the output elements of several functions there can be distinguished three different kinds of potential conflicts:

- The results of two lenses overwrite attributes of the same data elements of the data stage, referred to as a *write conflict*.
- One lens L_1 operates on data elements which attributes are modified by another lens L_2's output. In this case lens L_1 depends on L_2. This kind of conflict thus is a *dependency conflict*.
- If there are two or more lenses which have a potential *write conflict* between their results and a third lens uses these contested data elements as input, a *read conflict* occurs, i.e. an ambiguity about which of the two possible result sets the third lens operator should use.

Based on the assumption that the visualization pipeline itself (i.e. the visualization without applied lenses) is void of conflicts, a single lens will never result in a conflict as lens functions by design always take precedence over regular pipeline operators. Furthermore, read conflicts can only occur if there are unresolved write conflicts. Solving the triggering write conflict automatically cancels the read conflict as well. Dependency conflicts can also be solved easily by appropriately marshaling the execution order of lens operators. Only write conflicts can not be solved by intrinsic means.

However, before conflicts can be resolved they first need to be detected. To this end, formal description of the output elements of the lens operators is required. We chose to use a mathematical set notation as follows.

Each of the four stages in the DSRM is represented by a set A_i of attributes a_{ik}; $1 \leq i \leq 4; 1 \leq k \leq |A_i|$ that are associated with the data elements at the corresponding stage [4], i.e. A_1 correspond to *Raw data* attributes, whereas elements from A_2 represent attributes of *analytical abstractions*, and so on.

A lens operator f transforms data elements with associated attributes from a source set S_n^f (n^{th} stage) to a target set T_m^f (m^{th} stage):

$$S_n^f \subseteq \bigcup_{1 \leq i \leq n} A_i, \quad T_m^f \subseteq \bigcup_{m \leq i \leq 4} A_i$$

This formalism allows us to detect, and in some cases to automatically resolve, the aforementioned conflict types. Let

$$f : S_n^f \rightarrow T_m^f; 1 \leq n \leq m \leq 4, \text{ and}$$

$$g : S_p^g \rightarrow T_q^g ; 1 \leq p \leq q \leq 4.$$

A potential *dependency conflict* occurs if $T_m^f \cap S_p^g \neq \emptyset$. In this case g depends on f and therefore, to avoid dependency conflicts f must be executed before g. This guarantees attributes are only used as elements of an input set after all modifying operations have completed.

A *write conflict*, on the other hand, occurs if $T_m^f \cap T_q^g \neq \emptyset$. As mentioned above, although this type of conflict can be detected, it does not lend itself to an inherent solution.

However, smart lenses must have the ability to resolve even write conflicts automatically. One approach to address this problem is a priority mechanism that determines which lens operator takes precedence. To this end, we have extended our script in such a way that it includes priority values for operators.

4 Realization and Usage Examples

Given such a script that reflects the user's interests, we now want to illustrate the different aspects of smart lenses in the scope of the visualization system LandVis [22] for healthcare data. It shows a map divided into areas correlating to the administrative districts on various levels (federal states, districts, ZIP code areas). Different choropleth representations and icon techniques are available for the incident analysis of several diseases (e.g. flu or respiratory problems). The application of the major influence factors identified in Section 3.2 is demonstrated in each of the following examples. The goal in every case is to provide additional information about the data in the region of interest whereas the smart lens application provides the automatic adjustment of relevant parameters.

A common task when exploring geo-referenced data sets is the localization of extreme values. Moreover, the spatial location of this value can exhibit temporal dependecies – for LandVis, monthly aggregates of disease incidents have been recorded over several years –, so one may look for the *current* maximum at the current time step. Figure 3 shows a 'maximum lens' which could be used in this situation. The *lens function* modifies the fill color of the map region according

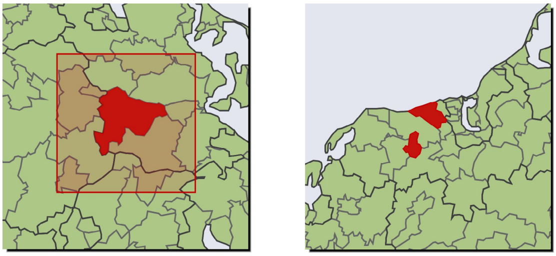


Fig. 3. Example 1: A 'maximum lens' modifies the fill color of map areas according to the number of incidents. The shape is either set to a specific shape (left), or data-driven (right), positioned automatically over the area with the current maximum.

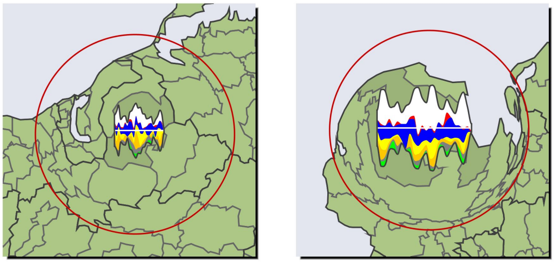


Fig. 4. Example 2: Lens that embeds ThemeRiver [23] icons into the map representation for detail analysis, combined with a fisheye distortion with a user-defined magnification factor. The lens is initially positioned over the user's state 'home' location.

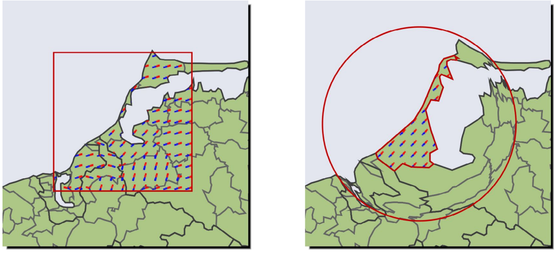


Fig. 5. Example 3: Interactively positioned lens for the analysis of temporal trends. The directional texture in this example consists of red-tipped indicators that point to the month of the yearly incident maximum (1 o'clock is January, 12 o'clock is December).

to the number of incidents by superimposing a full red color with increasing opacity. The *lens shape* can be either static according to the user's preference, e.g. rectangular (Fig. 3 left), or data-driven, i.e. adjust itself automatically to match the shape of the currently focussed map area (Fig. 3 right). The (initial) *lens position* is centered on the map area exhibiting current maximum value. Position updates can be data-driven as well, by automatically re-locating the lens when the current maximum changes.

A second common task is to observe a specific region of interest in high detail. An example is the local region where a physician's office is located and where therefore the temporal developement of treated diseases is of primary interest. In this case, most lens parameters will be determined in the script according to user requirements. Figure 4 gives an example for such a lens. The lens is initially positioned over the physician's stated location (address), while the map region under the lens is also enlarged using a fisheye magnification [15] with a user-specified magnification factor. The lens function was set to use a ThemeRiver [23] icon for temporal analysis.

As a last example, Figure 5 shows the support for an analysis of temporal trends in the disease data set. The lens function creates a choropleth representation in the lens region that uses a directional texture to indicate the yearly maximum of incidents for a given disease. Again, the lens shape can either be set to a fixed shape (user-driven, Fig. 5 left), or automatically adapt to the shape of the currently focussed map area (data-driven, Fig. 5 right). The right figure also shows an example of a two-lens combination, where a fisheye distortion enlarges small map regions to enhance the legibility of the directional texture. The lens position in both cases is manipulated exclusively through user interaction as the current region of interest can not be derived from the broad task specification.

5 Conclusion

In this paper, we proposed a new approach to define smart lenses, meaning that our approach allows the versatile definition and combination of arbitray lens functions for arbitrarily shaped regions of interest. This is achieved by using as the basis for lens plug-ins Chi's Data State reference Model, wherein lenses can be defined as operators on any stage of the DSRM. Lens regions are defined by means of half space intersection both within the data and the presentation domain. Possible conflicts of lens combinations are detected and partially resolved using a formal set notation.

Thus, single lenses can be employed to modify specific aspects of the visualization within the current RoI. To this end we proposed a systematization of the influence factors that guide the selection and adequate parametrization of those lenses. A major benefit of our approach therefore is that no extensive task model is required.

However, our case studies are only the first step. A user evaluation could provide valueable feedback on the relevance of the aspects considered and their effectiveness in improving the use of lenses. Also, the current solution could be improved in a number of ways. So far, the script-based definition of lenses is limited, especially regarding the operators constituting the lens function. An extension of the script mechanism to allow more flexibility and fine-grained control over operators seems a logical next step. Other aspects that could be improved are the lens prioritization used to resolve write conflicts and the automatic selection of scripts/lens functions based on the current task. So far, we use verbal descriptions of common goals (e.g. localization, identification, comparison, recognition of trends) in the script selection. However, a task model represents a structured task description, and therefore could further enhance the script selection process. Such task models have been used by our research group for the task-specific adaptation of visual interfaces on different platforms [19]. It would be worth to investigate how these task models can be used for automatic adaptation of smart lenses.

References

1. Ellis, G., Dix, A.: Enabling Automatic Clutter Reduction in Parallel Coordinate Plots. IEEE Transactions on Visualization and Computer Graphics 12(5), 717–724 (2006)
2. Andrienko, N.V., Andrienko, G.L.: Exploratory Analysis of Spatial and Temporal Data – A Systematic Approach. Springer, Heidelberg (2006)

3. Bier, E., Stone, M., Pier, K., Buxton, W., DeRose, T.: Toolglass and Magic Lenses: the See-Through Interface. In: Proceedings of the 20th annual conference on Computer graphics and interactive techniques, pp. 73–80 (1993)
4. Fuchs, G., Thiede, C., Schumann, H.: Pluggable Lenses for Interactive Visualizations. In: Poster Compendium of InfoVis 2007 (October 2007)
5. Chi, E.: A Taxonomy of Visualization Techniques Using the Data State Reference Model. In: IEEE InfoVis., pp. 69–75 (2000)
6. Griethe, H., Fuchs, G., Schumann, H.: A Classification Scheme for Lens Technique. In: WSCG Short Papers, pp. 89–92 (2005)
7. Ellis, G., Dix, A.: A Taxonomy of Clutter Reduction for Information Visualisation. IEEE Transactions on Visualization and Computer Graphics 13(6), 1216–1223 (2007)
8. Keahey, T.A.: The Generalized Detail-In-Context Problem. In: IEEE InfoVis., Washington, DC, USA, pp. 44–51. IEEE Computer Society, Los Alamitos (1998)
9. Fuchs, G.A., Holst, M., Schumann, H.: 3D Mesh Exploration for Smart Visual Interfaces. In: Proc. Intl. Conference on Visual Information Systems (2008)
10. IDELIX Software Inc.: Pliable Display Technology White Paper, `www.idelix.com`
11. Keahey, T.: Area-normalized thematic views. In: Proceedings of International Cartography Assembly (1999)
12. Loughlin, M., Hughes, J.: An Annotation System for 3D Fluid Flow Visualization. In: Proceedings of IEEE Conference on Visualization 1994, October 17-21, 1994, pp. 273–279, CP31 (1994)
13. Tominski, C., Abello, J., van Ham, F., Schumann, H.: Fisheye Tree Views and Lenses for Graph Visualization. IV, pp. 17–24 (2006)
14. Bier, E., Stone, M., Pier, K.: Enhanced Illustration Using Magic Lens Filters. Computer Graphics and Applications, IEEE 17(6), 62–70 (1997)
15. Rase, W.D.: Fischauge-Projektionen als kartographische Lupen. In: Salzburger Kartographische Materialien, vol. 26, pp. 115–122 (1997)
16. Leung, Y.K., Apperley, M.D.: A Review and Taxonomy of Distortion-Oriented Presentation Techniques. ACM Trans. Comput.-Hum. Interact. 1(2) (1994)
17. Kosara, R., Miksch, S., Hauser, H.: Semantic Depth of Field. In: IEEE InfoVis. (2001)
18. Tominski, C., Fuchs, G., Schumann, H.: Task-Driven Color Coding. In: Proc. 12th International Conference Information Visualisation (IV 2008), London, July 8 - 11, 2008. IEEE Computer Society, Los Alamitos (2008)
19. Biehl, N., Düsterhöft, A., Forbrig, P., Fuchs, G., Reichart, D., Schumann, H.: Advanced Multi-Modal User Interfaces for Mobile Devices - Integration of Visualization, Speech Interaction and Task Modeling. In: Proceedings 17th IRMA International Conference, Washington DC, USA (2006) (Short Paper)
20. Rathsack, R., Wolff, A., Forbrig, P.: Using HCI-Patterns with Model-Based Generation of Advanced User Interfaces. In: MDDAUI 2006 - Model Driven Development of Advanced User Interfaces, vol. 214 (2006)
21. Open Geospatial Consortium: OpenGIS Filter Encoding Implementation Specification, Version 1.1.0 (final). Open Geospatial Consortium Inc. (2005)
22. Schulze-Wollgast, P., Schumann, H., Tominski, C.: Visual Analysis of Human Health Data. In: IRMA International Conference (2003)
23. Havre, S., Hetzler, B., Nowell, L.: ThemeRiver: Visualizing theme Changes Over Time. In: IEEE InfoVis., pp. 115–123. IEEE Computer Society, Los Alamitos (2000)

Overlapping Clustered Graphs: Co-authorship Networks Visualization

Rodrigo Santamaría and Roberto Therón

University of Salamanca

Abstract. The analysis of scientific articles produced by different groups of authors helps to identify and characterize research groups and collaborations among them. Although this is a quite studied area, some issues, such as quick understanding of groups and visualization of large social networks still pose some interesting challenges. In order to contribute to this study, we present a solution based in Overlapper, a tool for the visualization of overlapping groups that makes use of an enhanced variation of force-directed graphs. For a real case study, the tool has been applied to articles in the DBLP database.

1 Introduction

Social network analysis has been a growing area of study in social sciences, mainly due to the amount of social information that can be recovered from internet social activities (blogs, chats, mail contacts, etc.) and from different public databases (movie and music databases, scientific article databases, etc.).

Information visualization has been extensively used by social scientists to aid in understanding these relationships. Most of the uses of information visualization in social networks are devoted to path-finding tasks, neighbor detection and most connected nodes (hubs) detection [9]. To perform these tasks, the usual visualization techniques are Node Links (NL) diagrams. NL diagrams represent entities as nodes (usually, points or small figures) and relationships as links (lines) that join related nodes. These diagrams are good for finding common neighbors and other characteristics, such as articulation points (nodes where two large subgroups join), but become cluttered and unreadable when the size of the graph is large [13]. Filtering and navigation through the graph must be implemented to dodge this problem. The primary alternate visualization technique to NLs are matrix graph representations, which perform well in finding most connected nodes and large complete subgraphs. Unfortunately matrix representations are not as good as NLs at conveying paths, and also have problems with large networks because of the space needed to represent the matrices (they are symmetric matrices, thus duplicating information, with lots of empty cells). The merging of both techniques is leading to promising results [12] although are still unable to deal with the large networks problem.

Some social data provides group information in addition to plain, individual relationships. This group information can help to simplify individual-level visualizations by taking them to group-level visualization, and it is key to understanding group relationships. In fact, research to find the best layout for NLs usually involves artificial

A. Butz et al. (Eds.): SG 2008, LNCS 5166, pp. 190–199, 2008.

classification of data by means of clustering or similar techniques, based on geometrical characteristics of the graph (usually path length between nodes). Most of these classification algorithms generate non-overlapping groups, which are useful for graph drawing but are not as good for group analysis, since real social groups are usually more complex, involving different degrees of overlap among groups.

There are techniques to find overlapping groups in data, such as fuzzy clustering [1] or biclustering [15], but there are also known overlapping groups in social data. Furthermore, some of the largest public databases contain information on social groups, such as IMDb (for movies) or DBLP (for scientific articles). In this paper, we present the application of a graph drawing method that speeds up the comprehension of these groups and exploits its use to simplify graph visualization. Section 2 presents related work in the area of social networks and clustered graph drawing. Section 3 explains the method to build overlapping group graphs, while Section 4 details its application to a real case study. Finally, Section 5 has our conclusions and summarizes some lines of future work that we are exploring.

Supplementary information, including more snapshots and a demonstration video with Smart Graphics' DBLP entries is available at http://vis.usal.es/artoverlapper.

2 Related Work

In this section, we will briefly survey the main social network tools and clustered graph drawing techniques in the present days.

2.1 Social Network Tools

There are a number of tools available to draw graphs, and the number of researchers that use these tools to analyze their data or as a starting point for their own graph implementations increase everyday. Some of these tools are mainly based in force-directed graphs, such as GraphViz [6] or Prefuse [11]. Force-directed graphs [8] make use of concepts such as gravitational or spring forces to layout nodes in a NL diagram.

These tools are usually more focused on aesthetics and usability, and they are used by a broad range of users, both scientific and non-scientific. For example, Prefuse, which is written in Java and it is accompanied by a comprehensive documentation and a large set of examples.

Other tools, such as Pajek [3] or JUNG (http://jung.sf.net) are focused on statistical analysis and drawing methods. Contrary to GraphViz or Prefuse, Pajek and JUNG are used mainly by specialists in social sciences and graph drawing.

2.2 Clustered Graph Drawing

Clustered graphs (CGs) are NL representations where groups or zones of related nodes are highlighted (specially colored, for example). These representations use non-overlapping clusters present in the data or obtained by clustering techniques (usually hierarchical clustering).

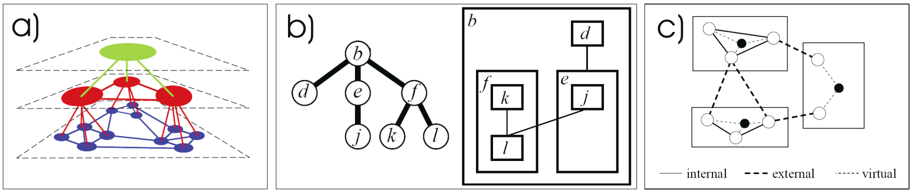


Fig. 1. a) Hierarchical clustered graph. A 3D visualization with different levels of clustering. Edges relate clusters together in the upper level. b) Compound graph, the cluster hierarchy at the left is translated to a graph layout of inclusion clusters. c) Force-directed clustered graph. Edges internal and external to clusters act as spring forces, with additional help of virtual edges by using virtual nodes in each cluster (all these figures are taken from [2]).

Three main types of CG drawings have been identified [2]: hierarchical clustered graphs, compound graphs and force-directed clustered graphs.

Hierarchical clustered graphs, introduced by Eades and Feng [5], start by drawing the highest level of a hierarchical clustering (only one cluster for all the nodes), and then go on drawing in decreasing z coordinates additional graphs with lower levels of clustering, where nodes are clusters and edges join clusters that were together in the upper clustering (see Fig 1a).

Compound graphs [19] are Hierarchical Clustered Graphs in which the inclusion relationship is taken into account to draw the hierarchical clustering in a single graph representation. The final visualization is very similar to a Treemap [18] (see Fig. 1b).

Finally, force-directed clustered graphs (FDCGs) are the most widespread Clustered Graphs. A combination of spring forces for a single clustering is used: inter-cluster, intra-cluster and (sometimes) ancillary forces by using virtual nodes in each cluster. In addition, a gravitational repulsion between each pair of nodes is applied (see Fig.1c).

Most of the social network tools discussed above have been used to implement FDCGs. For example, SocialAction [16] uses the Prefuse visualization kit and *betweeness* centrality (a measure of the relative importance of a node within a graph) to determine and draw clusters, simplifying the visualization of graph drawings. Vizster [10] is also based on Prefuse and group zones by clustering, allowing the user to define its granularity. Frishman and Tal [7] take GraphViz as a starting point for a dynamic drawing of clustered graphs. It is common in this implementations that the clustering displayed in the graph is a level of a hierarchical clustering, that can be changed at user's demand.

Besides Compound Graphs, where intersection between groups is reduced to inclusion, none of these graph drawing methods deal with overlapping groups, that are usually present in real data, or which can be obtained by newer classification techniques such as biclustering. Overlapping groups are usually a better way to display connections between entities, avoiding the threshold cut in hierarchical clustering that can assign doubtful nodes to a group.

3 Group Drawing with Overlapper

In this section we describe the drawing methods used by the presented visualization technique, focusing on graph building, layout and interaction.

3.1 Overlapper

Overlapper is a tool designed for visual analysis of data from different fields, such as social groups or biclustering results. It integrates and links different ancillary visualization techniques, such as parallel coordinates, scatter plots, tree maps and node-link diagrams to gain insight into the field of study. The main visualization in Overlapper is based in the graph that is described below.

The overall structure of the tool is redesigned for each field of study to fit with the specific characteristics of the data (e. g. node types, group types, relevant ancillary visualization techniques). Two versions have been developed, one for movie world analysis [20], awarded at the 14th Graph Drawing Contest [4], and another one for biclustering results analysis [17].

3.2 Group Building

Graph drawing with Overlapper centers on groups and their representation. To achieve this, the data to be represented should contain information on groups, either from previous information (as could be the case of databases like IMDb or DBLP) or from classification techniques (producing either overlapping or non-overlapping groups). No

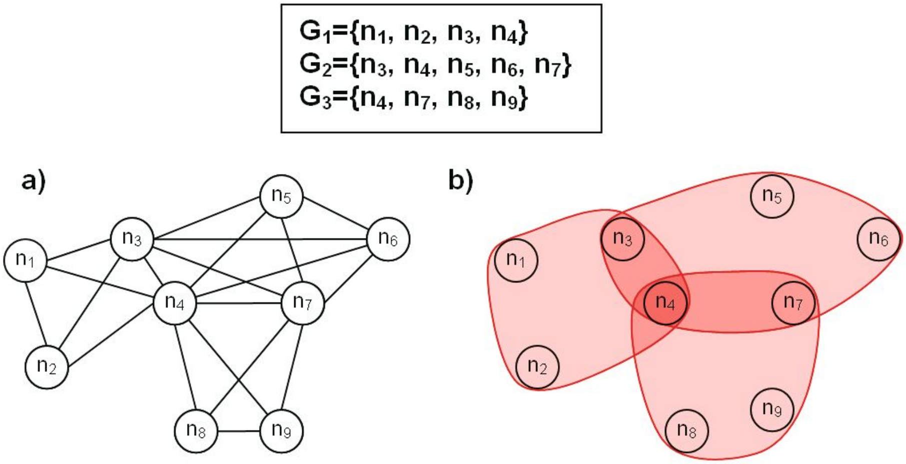


Fig. 2. a) Three groups are represented as complete subgraphs, with edges between all their members. b) Edges are hidden and replaced by transparent hulls wrapping the elements in each group. The relationships between groups arise quickly and elements like n_4, present in the three groups, are highlighted by hull overlapping.

matter what the source of data, we produce a list of groups $G_1,...,G_k$ each one containing elements, that will be treated as nodes in a graph: $G_i = \{n_{i1},...,n_{ik}\}$. Note that, for some groups, it is possible that $G_i \bigcap G_j \neq \oslash$.

From this list of groups, each G_i is represented as a complete subgraph, with the resulting graph being the union of all these subgraphs (see Fig. 2). Nodes and edges in this graph $G = (N,E)$ correspond to:

$$N = \{n_i | \; \exists \; G_k \; with \; n_i \in G_k\} \tag{1}$$

$$E = \{(n_i,n_j) | \; \exists \; G_k \; with \; n_i,n_j \in G_k\} \tag{2}$$

3.3 Graph Layout and Drawing

The graph is displayed using a force-directed layout, with two kinds of forces, in a way similar to other force-directed graphs, such as the ones implemented in the social network tools discussed in Section 2. A spring force S attracts nodes joined by an edge, while a gravitational force X repulses each node from every other node (see eq. 3). Both forces depends on the distances among nodes. The S force is kept stronger than X to avoid dispersion of groups. The overall result is that nodes in the same groups tend to be closer and are separated from nodes in different groups.

$$F_i = \sum_{(n_i,n_j)\in E} S_{i,j} + \sum_{n_j \in N} X_{i,j} \tag{3}$$

The layout is computed iteratively, so after each cycle, nodes are relocated depending on the applied forces, and forces are again recomputed for the new locations of nodes.

For each layout cycle, nodes are drawn as circles at their recomputed positions. In addition, for each group a rounded transparent shape (hull) is drawn, instead of drawing their edges. The contour of the hull is determined by the positions of the outermost nodes in each subgraph, that are taken as anchor points for a closed spline. The outermost nodes are computed on-the-run by checking the positions of the nodes in each group, and determining which are the ones in the periphery at each moment (those with the minimum or maximum x,y coordinates).

The transparency level of hulls is determined by the maximum number of overlapping groups in a determinate set of groups, n_max. If 0 is the transparency level of transparent colors and k is the transparency level of solid colors, the transparency of each hull is $(k - k_0)/n_max$. k_0 is a low value that keeps the maximum overlap from being fully solid.

Hull drawing, although based on edges, does not clutter the visualization. The transparency of hulls makes intersecting zones among groups more opaque, thus highlighting the highly connected, hub-like zones.

Finally, to boost comprehension of the relationships among groups, intersecting nodes are drawn as pie charts, with as many sectors as groups the node belongs to. This way, after getting used to our method, the analysis of group interactions becomes easy and unambiguous, and identifying the most connected nodes, thanks to transparent

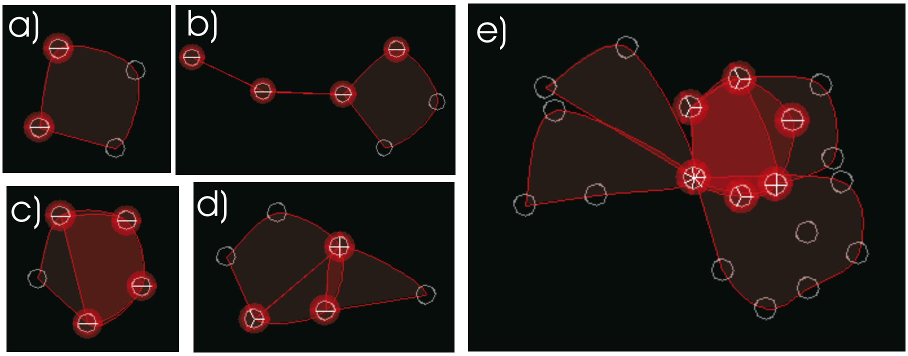


Fig. 3. Some examples of real group relationships from DBLP. Each hull is a group (an article) and each node an author. a) Three articles, in the first one four authors collaborated, the other two were written by single authors. b) A chain of scientific collaborations. Most left author has written two articles, one alone and another one with the chain's following author that, in turn, has written another article with the following author in the chain. Finally, this author also collaborated in a paper with three other colleagues, one of which has written an article alone. c) Two articles written by almost the same people except one person. d) Four articles. The most prolific author worked alone once, another time with just a colleague and two more times with groups of two and four other people. e) A more complex interrelationship of authors. The most relevant author, present in all (seven) publications is quickly identified. Also, the most prolific authors worked together on a couple of papers.

hulls and pie charts, is quicker (see Fig. 3). Note that for groups of two elements, hulls are drawn as lines, and for groups of one element, no hull is drawn at all, so in these cases piecharts are used to distinguish, for example, a member of a group that has worked in other projects, from a member that just worked in this group. These very small groups can occur in research paper datasets (some papers are authored by a small number of researchers).

3.4 Graph Interaction

Once the graph is built and displayed, it can be manipulated by the user in a number of ways. Regarding to the layout, the user can change the parameters X and S and modify the representation by dragging and fixing node positions. Regarding the graph drawing, the user can visualize or hide nodes, edges, hulls and pie charts; draw labels of node and group names; and highlight the nodes connected to a particular node. In order to facilitate the navigation through the graph, the user can search for author names, filter low related groups, and get an overview of the complete graph. Finally, the user can export the graph visualization to different image formats.

4 Case Study

In order to demonstrate the capabilities of our representation, we have used subsets of the DBLP article database [14]. Specifically, we have focused on the articles from

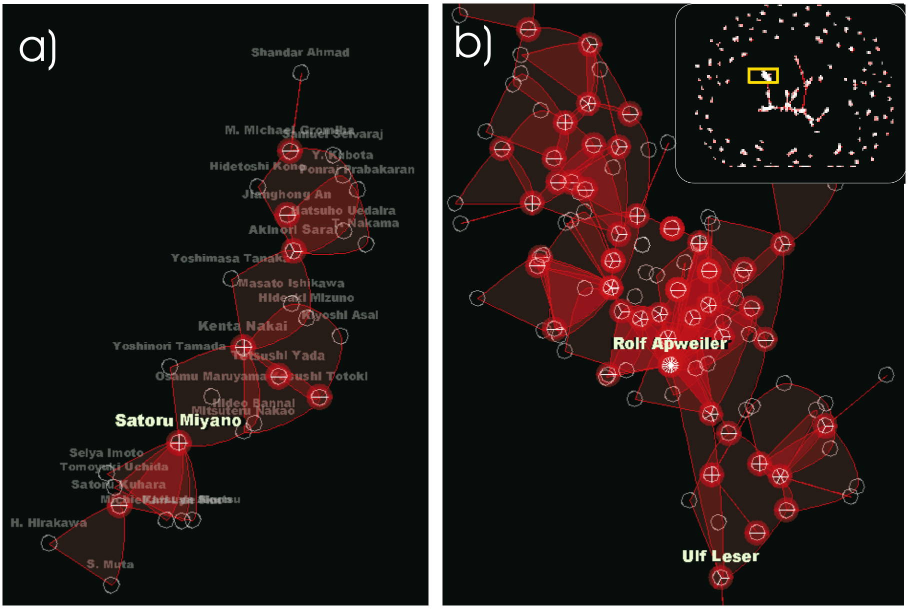


Fig. 4. a) Japanese researchers in a peripheral group of authors that have published in Bioinformatics. Although it is a more complex group than those presented in fig. 3, interactions between authors are clear. b) Top-left part of the central subgroup, where the most influent Bioinformatics' contributors are present. This zone is mainly populated by German authors. Top-right square shows the overall view of the complete graph, with disconnected, peripheral groups surrounding the central group, with clear branches.

the journal *Bioinformatics*, with 10 years of publications (since 1998) and over 3500 articles and 25000 authors.

The first five years of *Bioinformatics*, with 932 articles and 2197 authors, represents a good challenge for our visualization tool. Filtering all the articles not related to any other, we reduced the number of articles and authors to 615 and 1212, respectively. The graph layout, directed by forces, disperses unrelated articles around the visualization space, leaving the highest related subgroup in the center of the visualization (see Fig. 4b, top-right square).

It is remarkable that the nationality of the authors is reflected in the way research groups are formed and publish. For example, in Fig. 4a we observe that a large peripheral group is formed almost exclusively by Japanese researchers.

The central group, with the most influential authors in *Bioinformatics* in its first five years, also include nationality groupings (see Fig. 5a for Russian researches, interconnecting with German colleagues of Fig. 4b).

Although complexity in group interactions arises in the groups of Fig. 4 and Fig. 5 with respect to the ones in Fig. 3, relationships are still clear. Due to forces equilibrium, on few occasions one node can be placed too close to a group that contains nodes related to it, but the node itself does not pertain to the group. In these cases, pie charts

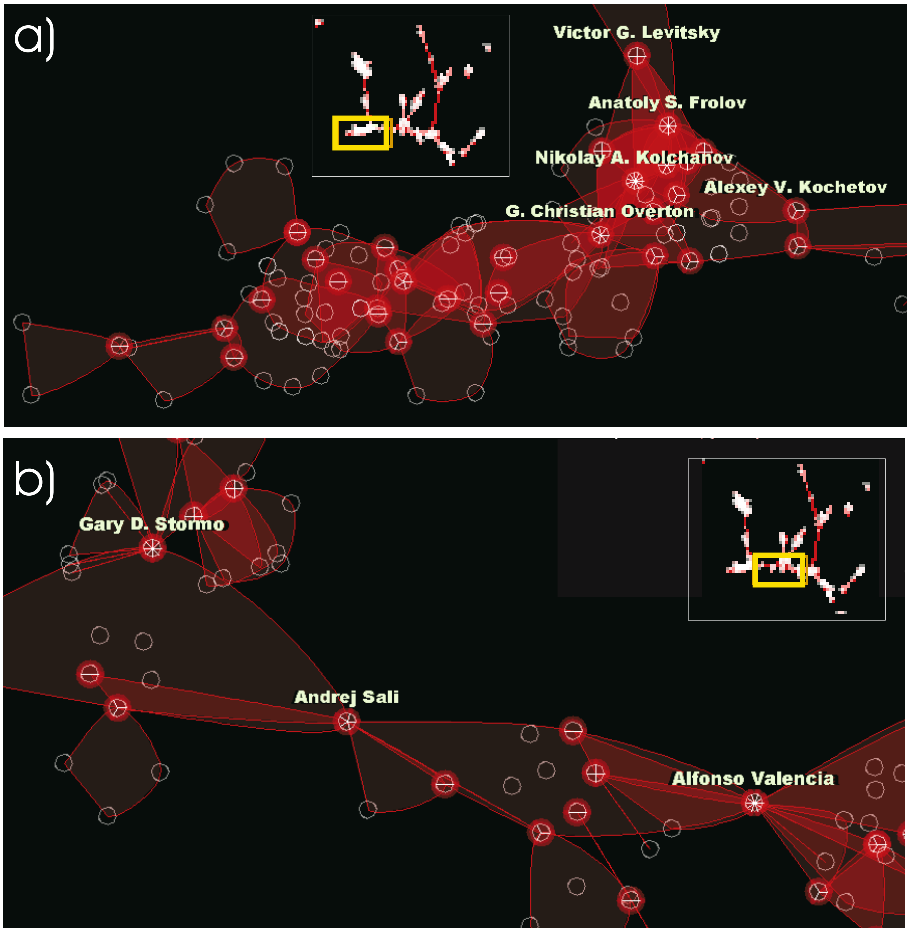


Fig. 5. a) Bottom left branch of the central main group. Most of the authors are Russian, with hub figures as Nikolay A. Kolchakov and articulation points as Victor G. Levitsky and Alexey V. Kochetov. b) Central part of the main group. Here, Alfonso Valencia reveals as one of the most connected nodes and also as an articulation point. Other articulated and connected authors are Gary D. Storno and Andrej Sali.

disentangle possible ambiguities. Hub nodes and articulation points are identified quickly, as can be seen in Fig. 5b. The identification of hub nodes is a difficult task in NL diagrams [12]; and it is mainly solved by this visualization.

Finally, we must consider that article graphs are usually sparse, and therefore cluttering is less frequent than in other denser graphs. However, the use of hulls instead of edges significantly simplifies the comprehensibility of the visualization whereas the traditional graph drawings of nodes and edges are unreadable even for these relatively simple examples (see Fig. 6).

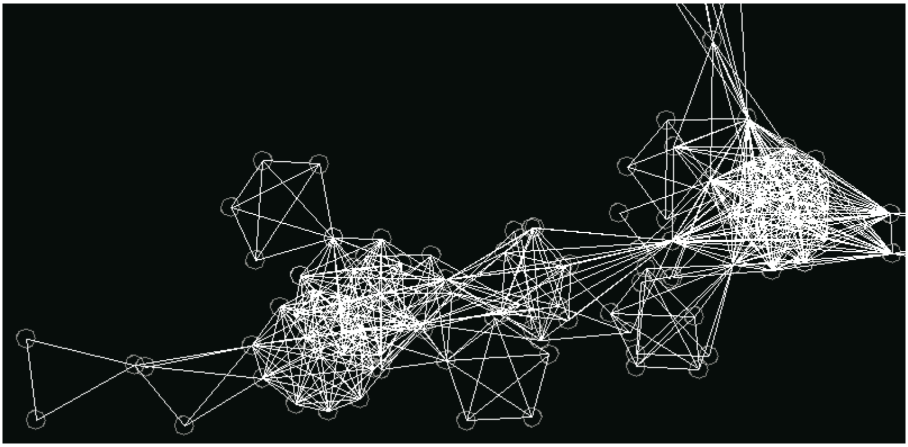


Fig. 6. Traditional NL representation of fig. 5a. The visualization becomes cluttered and details about groups are unreadable.

5 Conclusions and Future Work

We have developed a new graph drawing method based on a different way of representing relationships among nodes. Individual relationships are taken as the basic infrastructure for the graph layout, but are hidden to the benefit of group relationships. The resulting graph drawing method is only applicable if group information is present or can be inferred with some classification technique. In these cases, which are common in social networks, the overlapping clustered graph drawing has advantages over standard NL diagrams. The edge cluttering is avoided and is substituted by an unoffensive hull overlapping, that is exploited to highlight intersecting groups.

The presented case study demonstrates that our method can successfully deal with large sparse graphs without losing readability and provides quick insight into group relationships. Also, thanks to the pie charts and the force-directed graph layout, the identification of hub nodes and articulation points is enhanced.

The use of this technique in denser graphs is under research. The overlapping display method will need modifications in the layout algorithm to deal with possible misplacing of nodes inside group hulls in which they are not included. This issue is solved by the force-directed layout for sparse graphs, but becomes a problem in denser ones.

Acknowledgments

This work was supported by the Ministerio de Educación y Ciencia of Spain under project TIN2006-06313 and by a grant from the Junta de Castilla y León autonomous government.

References

1. Baraldi, A., Blonda, P.: A survey of fuzzy clustering algorithms for pattern recognition. IEEE Trans. on Systems, Man, and Cybernetics 29(6), 786–801 (1999)
2. Brockenauer, R., Cornelsen, S.: Drawing clusters and hierarchies. In: Kaufmann, M., Wagner, D. (eds.) Drawing Graphs. LNCS, vol. 2025, pp. 193–227. Springer, Heidelberg (2001)
3. de Nooy, W., Mrvar, A., Batagelj, V.: Exploratory Social Network Analysis with Pajek. Cambridge University Press, New York (2005)
4. Duncan, C.A., Kobourov, S.G., Sander, G.: Graph drawing contest report. Technical report (2007)
5. Eades, P., Feng, Q.-W.: Multilevel visualization of clustered graphs. In: North, S.C. (ed.) GD 1996. LNCS, vol. 1190, pp. 101–112. Springer, Heidelberg (1997)
6. Ellson, J., Gansner, E., Koustsofios, L., Stephen, N., Woodhull, G.: Graphviz – open source graph drawing tools. In: Mutzel, P., Jünger, M., Leipert, S. (eds.) GD 2001. LNCS, vol. 2265, pp. 483–485. Springer, Heidelberg (2002)
7. Frishman, Y., Tal, A.: Dynamic drawing of clustered graphs. In: Infovis, pp. 191–198 (2004)
8. Fruchterman, T.M.J., Reinhold, E.M.: Graph drawing by force-directed placement. Software – Practice and Experience 21, 1129–1164 (1991)
9. Ghoniem, M., Fekete, J.-D., Castagliola, P.: On the readability of graphs using node-link and matrix-based representations: a controlled experiment and statistical analysis. Information Visualization 4, 114–135 (2005)
10. Heer, J., Boyd, D.: Vizster: visualizing online social networks. In: IEEE Symp. on Information Visualization, p. 5 (2005)
11. Heer, J., Card, S.K., Landay, J.A.: Prefuse: a toolkit for interactive information visualization. In: SIGCHI Human Factors in Computing Systems, pp. 421–430. ACM Press, New York (2005)
12. Henry, N., Fekete, J.-D.: Matlink: Enhanced matrix visualization for analyzing social networks. In: Human-Computer Interaction, pp. 288–302 (2007)
13. Herman, I., Melançon, G., Marshall, M.S.: Graph visualization and navigation in information visualization: A survey. IEEE Trans. on Vis. and Comp. Graph. 6(1), 24–43 (2000)
14. Ley, M.: The dblp computer science bibliography: Evolution, research issues, perspectives. In: 9th Intl. Symp. on String Processing and Information Retrieval, pp. 1–10 (2002)
15. Madeira, S., Oliveira, A.: Biclustering algorithms for biological data analysis: a survey. IEEE/ACM Trans. of Computational Biology and Bioinformatics 1(1), 24–45 (2004)
16. Perer, A., Shneiderman, B.: Balancing systematic and flexible exploration of social networks. IEEE Trans. on Vis. and Comp. Graph. 12(5), 693–700 (2006)
17. Santamaría, R., Therón, R., Quintales, L.: Bicoverlapper: A tool for bicluster visualization. Bioinformatics 24(9), 1212–1213 (2008)
18. Shneiderman, B., Wattenberg, M.: Ordered treemap layouts. In: Infovis, pp. 73–78 (2001)
19. Sugiyama, K., Misue, K.: Visualization of structural information: automatic drawing of compound digraphs. IEEE Trans. on Systems, Man and Cybernetics 21(4), 876–892 (1991)
20. Therón, R., Santamaría, R., García, J., Gómez, D., Paz-Madrid, V.: Overlapper: movie analyzer. In: Infovis Confererence Compendium, pp. 140–141 (2007)

dream.Medusa: A Participatory Performance

Robyn Taylor[1], Pierre Boulanger[1], and Patrick Olivier[2]

[1] Advanced Man-Machine Interface Laboratory,
Department of Computing Science, University of Alberta
T6G 2E8 Edmonton, Alberta, Canada
{robyn,pierreb}@cs.ualberta.ca
[2] School of Computing Science, Culture Lab, Newcastle University,
Newcastle upon Tyne, NE1 7RU, UK
p.l.olivier@ncl.ac.uk

Abstract. We describe a system which allows several audience members to participate in a performance of an interactive media piece. The performance is created using Max/MSP and Jitter, and is controlled by live voice as well as by participant-operated manipulated objects. The performance was created as part of an interactive art exhibit exploring a night of dreaming and was devised in order to explore the experience of lucid dreaming. We discuss our experiences with the performance and the potential use of participatory performance as a vehicle for exploring wider issues in interaction design.

1 Introduction

In our previous work developing multimedia performance spaces, we have focussed primarily on performer driven interaction techniques, such as systems which allow a performer to interact with an animated character [9] and interactive video systems [8]. These systems are in essence structured around the performer, allowing only the performing artist to shape the development of the responsive performance. Our goal is to explore interactive media pieces in which participating audience members' contributions modify and manipulate the development of the ongoing performance. Previous artists have developed responsive creative environments which enable audience members to transition from passive spectators to active participants in a performance. These include Sheridan's responsive *iPoi* performances [7] and Winkler's sound and video installation *Light Around the Edges* [10].

dream.Medusa is an interactive media work which allows participants to interact with a responsive video system by manipulating specially created objects and exploring how their manipulations change a video visualization. Each performance of the work is necessarily unique, as the discovery and exploration process each participant undergoes as s/he learns to control the video environment using the control devices is different. The conceptual basis of *dream.Medusa* is the exploration of the experience of lucid dreaming [11]. Participants are provided with tools to interact with and control aspects of a simulated dream, represented by the responsive visual display.

A. Butz et al. (Eds.): SG 2008, LNCS 5166, pp. 200–206, 2008.

In addition to being a compelling and dynamic art form, we believe that participatory performance allows us an opportunity to explore human-computer interaction in a non-traditional setting. Participatory performance has the potential to assist us in our ongoing exploration of participants' interactions in performance environments. It is our hope that our observations will reveal context-sensitive insights into collaborative creative behaviour that could be integrated into the design of interactive systems for everyday use. After a describing *dream.Medusa* we discuss its role in an emerging programme of research which uses artistic practice to inform interaction design, and describe how our development strategy and project goals fit within that context.

2 The dream.Medusa Performance

In contrast to our previous work, *dream.Medusa* has a performance frame that incorporates not only the actions of a live performer – a singer – but also includes a group of four participants who help create the performance which the spectating audience observes. This characterization of roles in partipatory performance design was devised by Sheridan *et al.* [7] and as they have noted, each repetition of a partipatory performance is uniquely shaped by this interplay between performers, participants and spectators.

The singer sits in front of a video screen, four participating audience members join her in the 'staging' area, and the remaining spectators watch from a distance

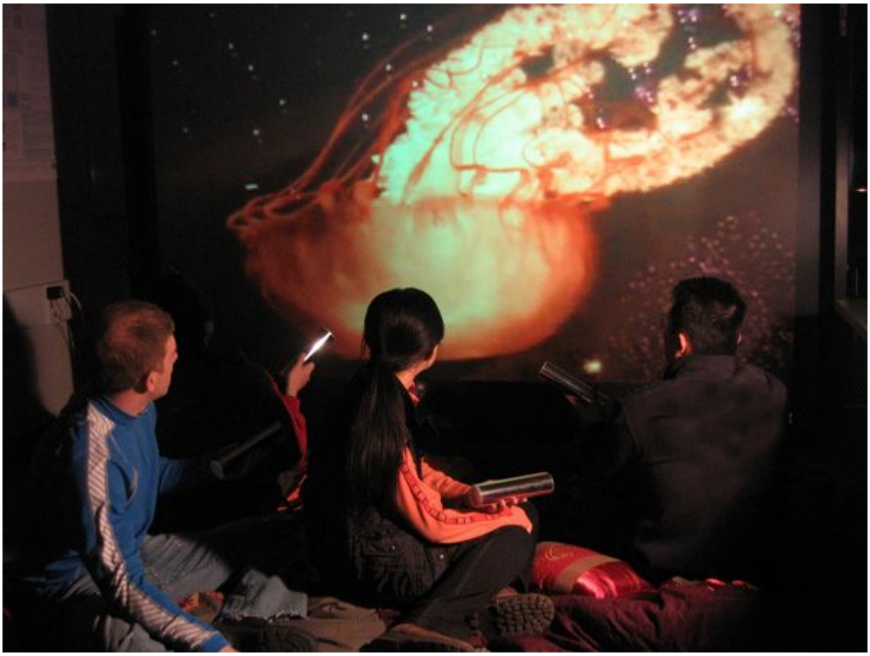


Fig. 1. Participants interacting with the responsive video environment

(see Figure 1.) During the performance, hypnotic visual images (videos of floating and drifting coloured jellyfish) are projected upon the large screen. As the singer vocalizes, her vocalizations are visualized through colour and imagery which is superimposed on the underlying video stream. The size of the screen, and the proximity of the participants to the display ensures that the visualization fills as much of their visual field as possible, helping immerse them in the audiovisual experience.

The four participants each hold a controller device with which they are able, at key points in the performance, to modify the playback of the video streams. The movement of each participant's controller modifies a different, mutually orthogonal aspect of the video playback. The combination of the singer's vocal manipulations and the interactions of the participants holding control devices determines the appearance of the video visualization. The participants are in no way practiced users of the system – they are audience members who are experiencing the performance for the first time while participating in its creation. They are instructed simply to explore the interactive objects and to try to learn how to control the visualizations so as to produce pleasing visual effects.

3 Voice Controlled Interaction

In order to allow the performer to interact with the video environment, her live singing is analyzed by the `fiddle~` object developed for Max/MSP [2] by Puckette *et al.*[6]. This object outputs the harmonic spectrum of the vocal stream. Using the same strategy developed to allow vocal input to be used as a control mechanism in our previous performance piece, *Deep Surrender*[8], each of the first three partials in the harmonic spectrum are mapped to the red, green and blue components of a Jitter video stream. This video stream is then superimposed over the underlying video footage; the resulting effect being that the singer's voice causes coloured images to appear and disappear on the screen as she sings.

Mapping the partials in the voice to the colours in this way allows the performer to manipuate the colour of the imagery by making subtle changes to her vocal timbre. This mapping is highly responsive and allows the singer to exercise fine control over the imagery by being carefully attentive to the video manipulations as she modifies her vocal tone.

4 Participant Control

In addition to allowing the singer to direct the development of the performance, we also allow four audience members to function as participants in the performance. We provide them with deliberately mysterious objects – mirrored tubes, which (unbeknownst to the participants) contain Nintendo Wiimote devices (see Figure 2.) Akamatsu's `aka.wiimote` plugin for Max/MSP[1] allows us to obtain orientation data from the 3-axis accelerometers contained in four Wiimote devices and use it to interact with the Max/MSP and Jitter environments.

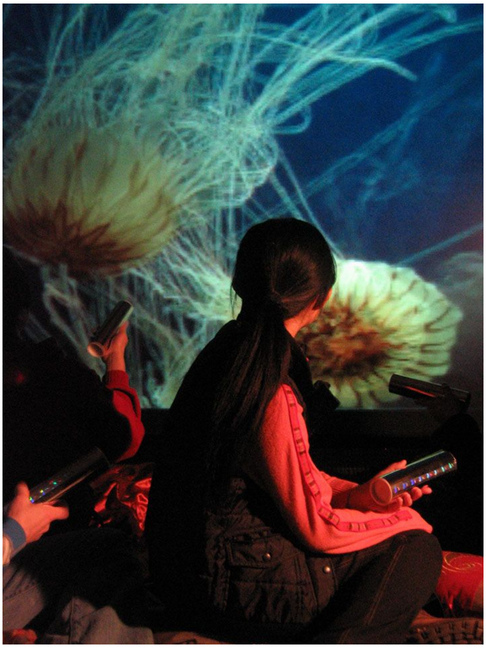


Fig. 2. A participant holds the controller object

Our participants are told that at various points in the performance their object will signal to them, via a pulsating "heartbeat" sensation, that it is activated. The participants are told that when their object is activated they will each be in control of an aspect of the video playback. They are not instructed as to how to control the object, rather they are encouraged to play with the object, manipulate it, and attempt to identify what aspect of the visualization they are controlling, and how.

When the participants move the control devices, the accelerometers contained in the Wiimote provide Max/MSP with readings indicating the orientation of the device. These orientations are then mapped to the control parameters of the video playback, so the participants can maniupulate the video parameters by waving and rotating the device. Each object is mapped to a different video parameter, so each participant has a different aspect of video playback to control. The four objects are mapped to colour balance, colour saturation, video blending, and edge detection. During the performance, different combinations of objects become activated simultaneously. This mapping also shifts through the course of the performance so the participants' attention must remain focussed in order to maintain understanding and control of how they are changing the video playback.

While the participants may simply focus on their own interactions and responsive parameters, they may also choose to deliberately work together in order to create pleasing visualizations. We have on occasion observed participants attempting to coordinate their actions with those of other participants in order to explore the visualization space, while at other times the group dynamic is such that the participants choose to function independently, each exploring the effects produced by his or her own controller object.

Neither approach is encouraged or discouraged – the participants are free to directly talk or communicate with one another, or with the performer. The physical set-up of the staging (a traditional staged format with the participants and performer separated from the audience, with the participants facing the screen rather than the audience) facilitates the ability of participants to talk quietly or make eye contact with one another if they so choose without feeling that their interpersonal interactions are being overly observed by the viewing audience.

5 Artistic Concept

The system was used to create a fifteen minute performance exploring the concept of lucid dreaming, titled *dream.Medusa*. In a lucid dream, a dreamer becomes aware that s/he is not awake, but rather s/he is dreaming [11]. Through that conscious realisation that reality is in fact fantasy, the dreamer becomes able to interact with the dream environment and can enact change in the dream world. Of course, as anyone who has experienced this phenomenon knows, control of a lucid dream can be fleeting, and the dreamer often either loses conscious control of dream events, or awakens entirely.

We attempt to explore this idea of conscious control of dreamscapes through our activation and de-activation of the participants' control devices. The participants are immersed in the dream (represented by the video projections and musical performance) and at certain points, become aware that they have control of their environment via the control devices. Symbolically, the appearance of the devices themselves (the mirrored tubes) refers to a classic technique described in lucid dreaming literature, whereby dreamers are encouraged to examine mirrored reflections in order to identify oddities which may signify that reality is in fact a dream [5]. The participants move in and out of control of the visualization, as they would move in and out of control in a lucid dreaming scenario, with the simulation of the dream environment facilitated by the large scale audiovisual display.

6 Performance History

dream.Medusa was presented at Toronto's 2007 Nuit Blanche festival, as part of an installation exhibit which took visitors through various stages of the human sleep cycle. Since then, it has been peformed in concert and workshop settings in Mexico City and Newcastle upon Tyne. Participants have included non-English speakers who communicated with the English speaking performance team via gesture and the rudimentary assistance of an informal translator in order to explore the performance scenario. In another occasion, the piece was performed by a team of professional dancers who after considering the possibilities and limitations of the interaction system chose to attach the controller devices to their arms in order to use larger gestures to manipulate the video environment. Performing with each of these varied participant groups presented new challenges, as well as produced different performative results. The participants in Mexico City included a small child and his mother which engendered a relaxed and spontaneous group dynamic, fostering a high level of inter-participant collaboration, while the more formalized dance team employed a different strategy for controlling the visual environment, using their experience and training in the art of modern dance to interpret and mimic the jellyfish visualizations while controlling the Wiimote interactors.

Discussions with participants indicate that they found the participatory experience to be rewarding. In most cases, participants have been able to control the interaction objects within the performance's timeframe, and they have told us that they found the visualization and musical components to be evocative and enjoyable. Individuals have told us that they felt a sense of responsibility and contribution to the development of the performance (one participant remarked that "up there, we are all in it together, so we have to make it look good!").

7 Participatory Performance and Interaction

It is our hope that the observations we make, and the experiences we have when collaborating with our participants to peform pieces like *dream.Medusa* could be applicable to future design of creative collaborative systems. Existing non-traditional approaches to interaction design such as Cultural Probes [3] which encourage participants to respond to designers using a variety of media including photography and audio recording, or the ViP method of product design [4] which encourages designers to imagine the desired emotional responses they would like their designed product to evoke represent strategies and experimentation grounded in aesthetic practice, and illustrate the value of conducting design research in ways which at first may seem indirect, but due to their open-endedness yield unexpected and previously unexplored design options and opportunities.

We believe that when designing creative collaborative systems, participatory performance can offer a non-traditional environment within which insightful observations could be made about participant interaction. Participants in an interactive performance have unique incentives which encourage them to engage in creative behaviour, such as the desire to appear competent in front of an observing audience [10] and the desire to provide to the audience an aesthetically pleasing performative experience [7]. These motivations are obviously different than those of participants undertaking activities in a traditional laboratory or field-study setting, and it is our hope that this will reveal alternative facets of their approaches to interaction with responsive systems.

We document performances of the piece via video and note-taking, and are compiling a body of observational and anecdotal data from our experiences. It is our goal that from the behaviour of our participants in this, and in future participatory performance projects, previously unexplored insights into participant interaction in collaborative creative spaces will be uncovered that will inform the design of interactive multi-user systems.

Acknowledgements

This work was funded by the Natural Sciences and Engineering Research Council of Canada, Alberta Ingenuity, and iCore. We would also like to acknowledge the support of Culture Lab and the School of Computing Science, Newcastle University, UK.

References

1. Akamatsu, M.: aka.wiiremote, `http://www.iamas.ac.jp/~aka/max/`
2. Cycling 1974. Max/MSP and Jitter
3. Gaver, B., Dunne, T., Pacenti, E.: Design: Cultural probes. Interactions 6(1), 21–29 (1999)
4. Hekkert, P., van Dijk, M.B.: Designing from context: Foundations and applications of the ViP approach. In: Designing in Context: Proceedings of Design Thinking Research Symposium, vol. 5, pp. 383–394 (2001)
5. LaBerge, S.: Prolonging Lucid Dreams. NightLight 7(3-4); The Lucidity Institute
6. Puckette, M., Apel, T., Zicarelli, D.: Real-time audio analysis tools for Pd and MSP. In: Proceedings of the International Computer Music Conference, pp. 109–112. International Computer Music Association (1998)
7. Sheridan, J.G., Bryan-Kinns, N., Baylss, A.: Encouraging Witting Participation and Performance in Digital Live Art. In: 21st British HCI Group Annual Conference, Lancaster, UK, September 3-7, 2007, pp. 13–23 (2007)
8. Taylor, R., Boulanger, P.: Deep Surrender: Musically Controlled Responsive Video. In: Butz, A., Fisher, B., Krüger, A., Olivier, P. (eds.) SG 2006. LNCS, vol. 4073, pp. 62–69. Springer, Heidelberg (2006)
9. Taylor, R., Boulanger, P., Torres, D.: Visualizing Emotion in Musical Performance Using a Virtual Character. In: Butz, A., Fisher, B., Krüger, A., Olivier, P. (eds.) SG 2005. LNCS, vol. 3638, pp. 13–24. Springer, Heidelberg (2005)
10. Winkler, T.: Audience participation and response in movement-sensing installations. In: Proc. of the International Computer Music Conference (December 2000)
11. van Eeden, F.: A Study of Dreams. Proceedings of the Society for Psychical Research 26 (1913)

The Visual Expression Process: Bridging Vision and Data Visualization

Jose Fernando Rodrigues Jr., Andre G.R. Balan, Agma J.M. Traina,
and Caetano Traina Jr.

Universidade de São Paulo, Instituto de Ciências Matemáticas e de Computação
Av. Trabalhador São-carlense, 400 - Centro - CP: 668 - CEP: 13560-970 - São Carlos, SP, Brazil
{junio,agrbalan,agma,caetano}@icmc.usp.br

Abstract. Visual data analysis follows a sequence of steps derived from perceptual faculties that emanate from the human vision system. Firstly, pre-attentive phenomena determine a map of potential interesting objectives. Then, attentive selection concentrates on one element of a vocabulary of visual perceptions. Lastly, perceptions in working memory combine to long-term domain knowledge to support cognition. Following this process, we present a model that joins vision theory and visual data analysis aiming at settling a comprehension of why graphical presentations expand the human intellect, making us smarter.

1 Introduction

Visual data analysis refers to the process of using visually mediated cognition in order to benefit from data; such cognition shall be driven intuitively and in elucidative manner. But how can vision mediate cognition? Why data visualization is potentially intuitive? In the next sections we put together a chain of concepts in order to delineate a coherent model that answers these questions.

The goal is to settle a perspective of visual data representations – as defined in Visual Analytics [15] – so that analytical insights can be achieved. Analytical assets are taken for granted as a necessary step in order to advance the mastery of any field of expertise. Abstract analysis allows for problems to be treated through smaller concepts leading to broader solutions and new hypotheses generation. In this line, here we address the needs pointed by Johnson *et al.* [6], who recommend the characterization of how and why visualizations work. Our broader goal is to develop principles for visual representations and, according to Card *et al* [3], such characterization is an initial step towards this goal. That is, first we must understand how visualizations enable cognition.

In Section 2 we build up a conceptual structure based on vision mechanisms. Following, in section 3, we trace the relation between vision and visualization. Section 4 carries a brief discussion. Section 5 concludes the paper.

2 Vision Mechanisms for Data Analysis

Neuroscience researchers state that, although the brain can perceive multiple visual targets simultaneously, it cannot process them in parallel. The solution is to restrict the

A. Butz et al. (Eds.): SG 2008, LNCS 5166, pp. 207–215, 2008.

consideration of the objects presented to the eyes. To do so, the mechanism of vision concentrates on small regions of the visual field, regarding single objects one after the other, a sequential process ruled by what is called *attention*.

2.1 Maps of Saliences

Attention solves the problem of overloading the brain, however attention casting is not so simple. Which regions/objects of the visual field should be considered as candidates for attention? In the early stages of vision processing, a set of characteristics found in a scene can potentially pop up to the eyes defining *maps of saliences*.

The idea of salience is to reinforce the neurons-mediated perception of the areas in the scene whose statistics of visual properties contrast with those of its surroundings. In Figure 1, we visualize the building of a map of saliences through an example; the figure demonstrates the framework proposed by Itti *et al.* [4].

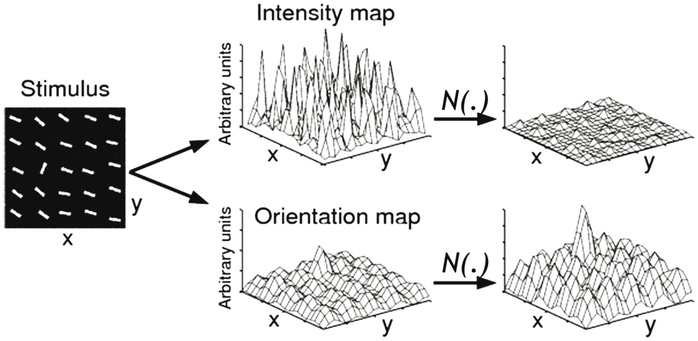


Fig. 1. The salience framework applied for color intensity and shape orientation. Illustration reproduced from Itti *et al.* [4].

After light stimuli reach the iconic memory (the primary short-term visual memory), they are processed in an early stage of the vision plexus. In parallel, this stage considers the whole visual field at multiple-scales of space and time. This first step produces a set of feature maps in the cortex area of the brain, each one summarizing how a particular visual feature is observed. For each map, a normalization operator $N(.)$ determines that several responses of similar amplitude generate maps without outstanding spots. At the same time, the normalization evidences the visual entities that contrast with their counterparts. The salience framework ends as it intersects all the generated maps onto a single panorama.

2.2 Attentive Selection

Once a map of candidates for attention is ready, it is necessary to choose one of them for processing. One at a time, the targets are considered according to the task to be performed (e.g., track all the red dots) or according to the data set (e.g., the blue dots pop up, what do they mean?). In the literature, the visual selection system has been widely assumed to have a pyramidal neuron structure. This model predicts a broad

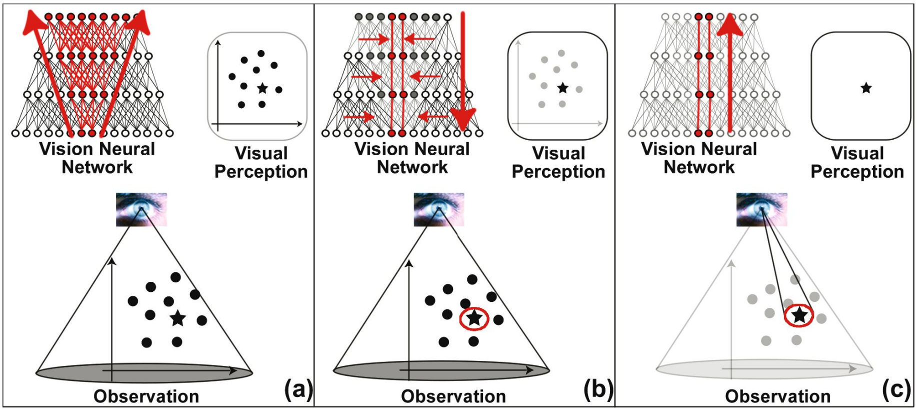


Fig. 2. Three-stage pyramidal visual selection

layer of neurons in its first level, narrowing down as it advances to upper layers, see Figure 2. The layers intercommunicate through feed-forward and feedback connections. According to Tsotsos [16], the pyramid of neurons performs three stages of processing:

◇ *bottom-up feed-forward*: the lowest layer receives stimuli from eye nerves, propagating the stimuli upward. The propagation generates a feed-forward inverted pyramid, see Figure 2(a). In this sub-pyramid, the activation of the connections and of the neurons depends on how the neurons are selective and biased. Biasing comes from upper layers according to the nature of the selection, task-oriented or data-oriented;

◇ *top-down feedback with WTA*: the upper layers return the initial stimuli through feedback connections downward the pyramid, Figure 2(b). At each layer of this reflexive propagation, a WTA (winner-take-all) process takes place fine tuning the network operation. Initially, at the top layer, WTA considers the entire visual field to compute the neurons with largest response, the winners at the first layer. The winners, then, promote a WTA among their input neurons at the layer immediately below, identifying the winners at this lower layer. At the same time, the connections that do not contribute to the winners are pruned (attenuated). This three-way process – WTA, descend and prune – layer by layer, repeats down to the lowest (input) layer. At the end, the winner at the output layer is traced back to its perceptual origin at the input layer;

◇ *bottom-up straight path*: the stimulus originating at the target object propagates through the path of winner neurons and of active connections, Figure 2(c). Now, the propagation proceeds without distracting stimuli, on each receptive field, as if the target was projected on a blank background.

2.3 Cognition, Memory and Vision

After a target is focused, it is potentially useful for cognition. Cognition refers to the acquisition or use of knowledge [3], a process assisted by the memory system. The relation between memory and cognition is studied in works on *Cognitive Architectures*, such as ACT-R and Soar. A widely accepted cognition-memory relationship is enunciated by the *Soar* architecture: memory refers to mechanisms that maintain and provide access to information created or retrieved during the performance of a task. Any computationally complete system must support such functionality, because computation is inherently a process that requires the temporary storage and manipulation of partial and intermediate products [19].

The *structural configuration* of memory, in Soar and other theories, more or less reflects the model described by Baddeley and Hitch [1]; working memory is comprised of three components: the central executive module, the phonological loop and the sketchpad. The central executive module determines the attention focus, guiding the visual perceptual system as, for example, by biasing the pyramidal selection mechanism. The phonological loop stores information related to sound, for example, a phone number that must be mentally repeated to prevent it from being forgotten. The sketchpad (also known as Visual Short-Term Memory – VSTM) is associated to the maps of saliences presented in section 2.1, storing information related to space and to visual features.

In the theory of working memory, VSTM is the main component in supporting cognition. Luck and Vogel affirm that people can retain only four to five single features in VSTM [10]. This is a problem because the greater the capacity of an individual's memory, the more information she/he has available for solving problems [7]. Hence, VSTM would limit cognition. However, according to Jiang *et al.* [5], the sense of vision is used as an extension of the working memory through VSTM. Even more, Jiang *et al* argue that visual representations need not to carry details simply because one can always rely on the outside world as her/his visual memory. This fact is crucial not only for the interaction of human beings with the physical world, it also explains the importance of computer graphics for data analysis and reasoning. That is, knowing that memory, although limited, is the factor that supports cognition, then, graphical externalizations appear as a natural ally for cognitive abilities; this is because graphical externalizations expand memory capacity.

3 The Visual Expression Process

We have reviewed the mechanisms of maps of saliences, attentive selection and vision-supported cognition. Now, we use this knowledge to analyze how data visual representations work. We introduce a model that links vision and visual data analysis, a theme overlooked in the visualization literature. Our framework is named Visual Expression Process, it encompasses three components: pre-attentive stimuli, visual perceptions and interpretations.

3.1 Pre-attentive Stimuli

Pre-attentive stimuli are the constitutional features that impel maps of saliences. A well-designed representation will present a high overlap between its map of saliences and its (implicit) map of semantical relevance. Although visual expression is not restricted to pre-attention, the design of visual representations is supposed to maximize design-specific pre-attentive effects. In consequence, there are many graphical properties, but

Table 1. Classes of pre-attentive features from the perspective of data representation

Pre-attentive classes	*Features*
Position	1D/2D/3D position, stereoscopic depth
Shape	line, area, volume, form, orientation, length, width, collinearity, size, curvature, marks, numerosity, convex/concave
Color	hue, saturation, brightness and texture
Animation	movement and intermittence

just a limited number of them are used for visual analysis [3]. More incisively, Ware [17] states that understanding what is processed pre-attentively is probably the most important contribution that vision science can bring to data visualization.

The features that are potentially pre-attentive span to a large set. We present an ideally exhaustive set of these features in Table 1. The classification we introduce is reductionist, nevertheless, the literature presents evidence that our generalization is coherent. About *color and texture*, Watt [18] affirms that, just like texture, color is the psychological response to the spectral characteristics of a surface; and that, different surfaces are perceived as having different colors. This is specially verifiable for visual data representations. Motter [12] observes that, early in visual processing, the incoming information is sorted and grouped according to the similarity of simple *shape features*, such as *orientation* or *size*, and of surface features, such as *color*, *luminance*, or *texture*.

Pre-attentiveness usually refers to physical properties of particular visual stimuli. Meanwhile, *salientness* refers to the attentional results of a scene as a whole. Each conception has its adequacies and they do not cancel each other. Either way, the fact is that when a set of visual features pops up *pre-attentively*, the brain builds a *map of saliences* of the scene.

"Effective visualizations must be composed of features that are potentially, and desirably, *pre-attentive*."

3.2 Visual Perceptions

Pre-attentive features provide maps of potential targets; following this stage, the next vision mechanism is attentive selection. Biased by user intention, one in many of the prominent areas in a visualization will monopolize the neuronal chain of vision. In what ways the targets of attention are perceived in a visual design?

We have tracked the possible ways in which visual manifestation occurs when the goal is data representation. We have extensively inspected the literature and found a limited set of possibilities in disjoint works. Here we put these findings together in a *visual vocabulary* whose elements appear recurrently over visualization techniques. We call these elements *visual perceptions*.

Visual perceptions are the traits that any user attentively seeks for in a visual representation. This notion is evident when they are not found and the pipeline outlined in Figure 3 is broken, preventing visual expressivity. We have observed that each visual perception is fired by one or multiple pre-attentive stimuli, which are discrete or continuous. According to the model we propose, visual perceptions occur after pre-attention

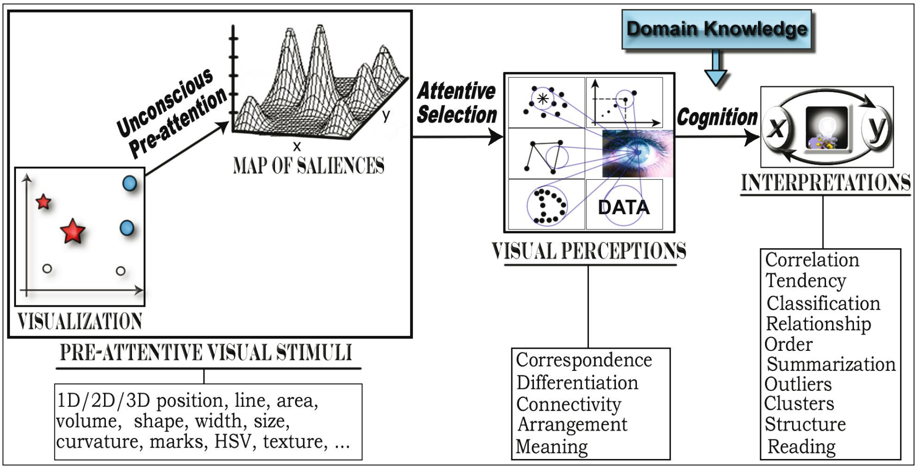


Fig. 3. The Visual Expression Process for visual data analysis

and before data interpretation, independently of the data domain. They bridge vision and data examination.

Our investigation indicates that the elements used in the second stage of visual expressivity are *correspondence*, *differentiation*, *connectivity*, *arrangement* and *meaning*. The most explored and verified of these phenomena, *correspondence* and *differentiation*, are noted by Bertin [2] and by Card *et al.* [3]. The third visual perception is presented by Mackinlay [11] who states that the notion of relationship among graphical entities comes from the perception of *connectivity*. Meanwhile, *Arrangement* arises from group positional configurations, largely studied by the Gestalt psychology [9]. The last perception is *meaning*, that is, resemblance to previous knowledge and/or expertise, studied in psychological models. These five abilities seem to be natural to human beings, more specifically:

- *correspondence*: Each position/shape/color has a direct correspondence to a referential map that is part of the scene (explicit) or that is mental (implicit). Explicit maps include axes, geographical maps, shape/color dictionaries, and position/shape/color ranges. Implicit maps include known orderings and shape metaphors;
- *differentiation*: Each position/shape/color discriminates graphical items. Differentiation is a correspondence in which the user creates a temporary map in memory;
- *connectivity*: Shapes, mainly edges, that convey information about relationships;
- *arrangement*: Gestalt principles of organization. Positional placements (closure, proximity and symmetry) that convey perception about group properties, for example clusters and structural cues;
- *meaning*: Positions/shapes/colors whose decoding comes from the expertise of the user or from previous knowledge. Meaning is a correspondence established from visual entities to concepts retained in long-term memory.

"Although visualization researchers have worked on the specificity of different designs [14], apparently, techniques of any nature explore means to express the limited vocabulary of *visual perceptions*."

3.3 Interpretations

The last step of the Visual Expression Process is to generate interpretations, the goal of visual data analysis. Interpretations refer to deductions, inferences or conclusions created when *visual perceptions* are considered in the context of the *data domain*.

The working memory, the long-term memory and the sensorial system support the generation of interpretations. Working memory has a limited capacity with only three to seven positions (chunks) to be occupied simultaneously and to support cognition. It can be filled with data from the long-term memory or from the sensorial system. However, maintaining data from the long-term memory in the working memory, for more than 3 seconds, demands cognitive resources. Differently, sensorial data is maintained in working memory without demanding cognitive resources; nonetheless, it asks for an effort of attention (sustained attentive selection).

The visual sensorial system, in turn, provides spatial information at rates much higher than the manipulation of images previously stored in long-term memory [17]. Besides, the access of the vision system to the working memory is so rapid that it rivals to the transferring of information from the long-term to the working memory. Filling the working memory with data, either from the long-term memory or from the sensorial system, takes between 100 and 250 *ms* [8].

Visual stimuli, assisted by visual perceptions, work similarly to the images stored in long-term memory. If data interpretation directly relates to cognition, then, *computer graphics make us smarter*. Interpretations are organically semantical; a suggestive set of these phenomena includes: correlation, tendency, classification, relationship, order, summarization, outlier, cluster, structure and reading.

"*Visual perceptions*, when in short-term memory, are conjoined with long-term domain knowledge to produce data interpretations. Indeed, successful *interpretations* necessarily emerge from the vocabulary of *visual perceptions*. For this reason, new principles for design could be investigated from a perceptual perspective."

3.4 The Process of Visual Expressivity

Figure 3 presents the Process of Visual Expressivity, which describes how visual representations can promote data investigation. It starts from pre-attentive stimuli and, with the aid of visual perceptions, it completes as data interpretations are achieved. Pre-attentive features of position, shape and color relate to raw data. Through the vocabulary of visual perceptions, interpretations are achieved. Interpretations relate not to data, but to new information in the context of the application domain. These three facts relate to vision mechanisms of saliences perception, of attentive selection over interesting targets, and of memory-based cognition.

4 Discussion

The Visual Expression Process breaks visual data analysis into its discrete constituents in order to explain why visualizations work from a visual-perceptual perspective. Given the fact that the Visual Expression Process derives directly from concepts of the vision system, we believe it is the underlying course supporting visualization techniques.

Visualizations unexceptionally depend on a small vocabulary of perceptions independently of the data domain. Visual perceptions define a distinct concept that, we believe, can promote a better support for consciousness in analyzing visual representations. This thesis is justified by the properties of visual perceptions, that:

- are common to every visualization, in contrast to interpretations;
- are not numerous, in contrast to the amplitude of visual patterns that manifest through position, shape and color;
- can be categorically related to the pre-attentive stimuli, allowing for immediate recognition and association;
- constitute a key element for visual analytical cognition.

Such comprehension can foment new approaches to the science of visual representations. In [13], we present a taxonomical analysis that classifies visual representation techniques according to how they employ position, shape and color. In another work [14], we concentrate on the process of spatialization as the determinant for data encoding; the work introduces an alternative design space and a general design model based on multiple cycles of spatialization. Complementing these previous papers, in this work we present evidences from the vision science to support the concepts that constitute the Visual Expression Process.

5 Conclusions

We have traced the connection between *vision science* and *visual representations* from the perspective of *data analysis*. Our model, named Visual Expression Process, departs from pre-attentive features and culminates on data interpretations; it has, as its core, a vocabulary of visual perceptions.

The text formulates a piece of communication intended to disseminate vision theory among visualization researchers. We teach concepts to furnish thoughts on how to develop principles for visual representations based on the notion of how they work. The goal is to set a perspective of visual data analysis in which smaller concepts transcribe existing problems, reaching solutions and generating new hypotheses.

Acknowledgements. Fundação de Amparo à Pesquisa do Estado de São Paulo (FAPESP), Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq) and Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (Capes).

References

1. Baddeley, A.D., Hitch, G.J.: Working memory. The psychology of learning and motivation: advances in research and theory 8, 47–89 (1974)
2. Bertin, J.: Graphics and Graphic Information-Processing, 273 pages. Walter de Gruyter, Berlin (1977/1981)

3. Card, S., Mackinlay, J., Shneiderman, B.: Readings in Information Visualization: Using Vision to Think. Morgan Kaufmann, San Francisco (1999)
4. Itti, L., Koch, C., Niebur, E.: A model of saliency-based visual attention for rapid scene analysis. IEEE Trans. on Pattern Analysis and Machine Intel. 20(11), 1254–1259 (1998)
5. Jiang, Y., Olson, I.R., Chun, M.M.: Organization of visual short-term memory. Experimental Psychology: Learning, Memory, and Cognition 26(3), 683–702 (2000)
6. Johnson, C., Moorhead, R., Munzner, T., Pfister, H., Rheingans, P., Yoo, T.S.: NIH/NSF Visualization Research Challenges, 36 pages. IEEE Press, Los Alamitos (2006)
7. Just, M.A., Carpenter, P.A.: A capacity theory of comprehension: individual differences in working memory. Psychological Review 99, 122–149 (1992)
8. Kieras, D.E., Meyer, D.E.: An overview of the epic architecture for cognition and performance with application to human-computer interaction. HCI 12, 391–438 (2007)
9. Koffka, K.: Principles of Gestalt psychology. In: International library of psychology, philosophy and scientific method. Harcourt, Brace and Company, New York (1935)
10. Luck, S.J., Vogel, E.K.: The capacity of visual working memory for features and conjunctions. Nature 309, 279–281 (1997)
11. Mackinlay, J.: Automating the design of graphical presentations of relational information. ACM Transactions on Graphics 5(2), 110–141 (1986)
12. Motter, B.C.: Neural correlates of attentive selection for color or luminance in extrastriate area v4. The Journal of Neuroscience 14(4), 2178–2189 (1994)
13. Rodrigues Jr., J.F., Traina, A.J.M., Oliveira, M.C.F., Traina Jr., C.: Reviewing data visualization: an analytical taxonomical study. In: International Conference on Information Visualization, Londres, UK, pp. 713–720. IEEE Press, Los Alamitos (2006)
14. Rodrigues Jr., J.F., Traina, A.J.M., Oliveira, M.C.F., Traina Jr., C.: The spatial/perceptual design space: a new comprehension for data visualization. Information Visualization 6, 261–279 (2008)
15. Thomas, J.J., Cook, K.A. (eds.): Illuminating the Path: The Research and Development Agenda for Visual Analytics. IEEE Press, Los Alamitos (2005)
16. Tsotsos, J.K.: The selective tuning model. In: Visual Attention Mechanisms, pp. 239–250. Kluwer Academic/Plenum Publishers (2003)
17. Ware, C.: Information Visualization: Perception for design, 486 pages. Morgan Kaufman, San Francisco (2004)
18. Watt, R.J.: Some speculations on the role of texture processing in visual perception. pp. 59–67. Massachusetts Institute of Technology, Cambridge (1995)
19. Young, R.M., Lewis, R.L.: The soar cognitive architecture and human working memory, pp. 224–256. Cambridge University Press, Cambridge (1999)

Flux: Enhancing Photo Organization through Interaction and Automation

Dominikus Baur, Otmar Hilliges, and Andreas Butz

University of Munich, LFE Media Informatics, Amalienstrasse 17,
80333 Munich, Germany
{dominikus.baur,otmar.hilliges,andreas.butz}@ifi.lmu.de

Abstract. In *Flux*, an application for digital photo collections, we provide methods for organizing and sharing photos in a convenient manner based on real-world physical interaction on a tabletop system. Additionally, the system supports four different orders based on quality, time, similarity and a combination of the latter two. The problem of scalability, which is especially relevant for a real-world sized photo collection is tackled with a combination of manual hierarchical clustering by the user and the automatic construction of "piles" of similar photos (depending on the currently active order) by the system. In this paper, we present the design goals, the visualization and interaction techniques deployed in *Flux*.

1 Introduction

With the advent of digital photography the number of photos in a collection radically increased: Former hindrances for taking photos like costs for development and material and the effort to go to a photo laboratory no longer existed which lead to an abundance of perceived photo opportunities. Still, although the new hardware provided the almost unlimited taking of photos, the corresponding software did not evolve equally fast: Even a hobbyist photographer still has to invest a lot of her time into organizing and pruning her collection in order to ever retrieve a photo again, a situation which in turn leads to the spread of more and more digital "shoeboxes" - akin to their real-world counterparts in being completely unsorted and never looked at again. Still, digital photos have one clear advantage compared to physical ones: They are much easier to endow with metadata that also enables machines to work with an otherwise inaccessible set of pixels.

In this paper we argue that the organization of digital photos can be enhanced by using automatic classification and analysis of images, but only if we also tightly integrate it with fitting interaction concepts. With Flux we created a system for photo collections that provides a natural, touch-based interaction on a tabletop display and relies on a physical metaphor to lower gateway hurdles. Automation and interaction are coupled to show as many photos as is conveniently possible and also let the user focus on certain aspects of the collection.

A. Butz et al. (Eds.): SG 2008, LNCS 5166, pp. 216–223, 2008.

2 Flux

2.1 Motivation

The organization of a digital photo collection is a task that grows in complexity and necessary effort with its size. Existing software solutions (see below) only marginally make use of the possibilities of automation and completely rely on the user's ability to categorize and organize the objects manually. This approach is reasonable in so far as the analysis of, for example, visual features still has its limitations: It only works on very low levels like, for example, color or edge distribution, while more sophisticated methods like face recognition are still too unstable under general conditions. But even on such a low level, a multitude of features can be extracted and have to be combined to arrive at one final similarity value which leads to the problem of choosing the right combination and weighting of different features. Additionally, the notion of similarity exists on multiple dimensions and levels of abstraction - while one person might, e.g., find two photos similar because of their bluish color schemes, another one might find them completely unrelated because of their showing different situations or persons. It is highly user dependent and thus should not be delegated to the machine, because especially in the context of a personal photo collection the user is the one to ask (as he also has to be able to retrieve a photo again later). Still, we think that the opportunities given by an automatic analysis can be gathered by coupling it closely with fluid interaction techniques to combine the knowledge of the system with that of the user. In this respect it would be best to let the machine do on its own what it will accomplish successfully or what would be too much bother for the user, then allow the user to adjust the result in a convenient way. To render this interaction more fluid and closer to its real-world counterpart we opted for a physical metaphor and direct manipulation without additional tools. Consequently, we chose a tabletop system as our platform which also corresponds to the typical setting where people work with real photos.

2.2 Related Work

Many applications for organizing photos exist for the hobbyist photographer. They mostly provide an organization scheme borrowed from the operating system's file system and allow for the creation of virtual albums or other basic organization and browsing facilities and searching based on keywords or filenames (Apple iPhoto[1], Google Picasa[2]). A common obstacle in this regard is the size of a collection: Because of practically unlimited storage space the number of photos is easily in the thousands which makes an organization based on the file system hierarchy tediously to navigate and labor-intensive to maintain. Two solutions from the academic world in this regard are a navigation based on either panning and zooming ([1]) or a semantic zoom ([2]). Even more photos can be displayed if redundant information is removed by automatically grouping similar objects,

[1] www.apple.com/iphoto

[2] picasa.google.com

either based on time or low-level features. Examples for the time-based approach are [3], Apple Aperture[3] and Adobe Photoshop Lightroom[4]. An evaluation of the second concept of analysis of low-level features (which uncovered inherent problems) was performed by Rodden et al [4].

Tabletop photo applications are mostly designed around a certain topic, for example, visualizing and working with a whole lifetime of annotated media ([5]) or providing tangible interaction with photos on a hybrid interface ([6]). Two examples for a physical metaphor in computer interfaces are [7] and [8], the former in easing the inhibitions of senior citizens in working with digital photos and the latter in enriching the classical desktop metaphor with physical objects. Flux takes ideas from these examples and tries to provide a convenient environment to work with a personal photo collection. Interaction techniques are based on a physical metaphor of photos and scalability is provided by combining automatic orders with quickly performable clustering.

2.3 Design

Our main goal for Flux was improving the organization of digital photo collections. This task is throughout the literature commonly referred to as *filing* (e.g., [9]) and relies on several other actions: The user has to be able to somehow sift through his collection to get an overview of the content and find certain objects. A purely search-driven approach is not feasible within the highly visual domain of photos, because finding a photo using text is only possible with meaningful keywords that have to be manually added by the user, a task that is only reluctantly performed, if at all ([10]). We therefore chose a browsing mechanism that lets the user quickly narrow in on a section of the collection and then linearly go through the contained photos. This “narrowing-in” has, of course, to be backed by some kind of hierarchical organization which is exclusively created by the user via easy gestures. Automatic analysis can be tapped in this regard by telling the system to arrange the photos according to a certain aspect (time, similarity), but this only works as a support for the user: We did not use automation-driven organization because of the downsides mentioned above. In this regard, scaling becomes an important aspect: If a collection is largely unorganized, many photos appear on the same level and have to somehow be visualized on a limited space. To relieve the user of this task, the system relies on the current order and forms “piles” of similar (visual or temporal) photos, volatile groups that reduce the cognitive load of the user and allow showing more pictures on less space.

Closely related to organization is the deletion of failed or unwanted photos, which Flux supports again in a two-part fashion: The system is able to distinguish low-quality photos based on certain features and can arrange them accordingly, but the final decision whether to delete those is left with the user. Aside from the organization facilities, Flux was also built with sharing in mind, i.e., showing photos to another person, which is especially relevant in a tabletop

[3] `www.apple.com/aperture`

[4] `www.adobe.com/products/photoshoplightroom`

context. Therefore, quick rotation and scaling are incorporated. Still, Flux is mainly aimed at a single user.

2.4 Overview of Flux

Flux (see figure 1) is a tabletop application for the visualization and organization of photos. Before launching the actual application the extraction of low-level features is performed to produce similarity and quality values. The initial screen shows the complete photo collection arranged by time (the default order). Photos can be combined into hierarchical clusters with a circular gesture. Overlaying workspaces can be created by an attached gesture and contain an arbitrary part of the whole collection. Objects on these workspaces can be more freely manipulated and act like physical objects but still provide the same options for interaction as the more restricted background. The system can sort all photos on a workspace or the background and afterwards automatically builds piles from related photos to save space and reduce clutter. Photos and workspaces are translated, rotated and scaled with two easy gestures.

Visualization: All objects in Flux (see figure 2) belong to a hierarchy (from bottom to top): **Photo objects** have a uniform appearance to make them readily recognizable. **Pile objects** are system-generated groups of similar photos. **Clusters** are user-created groups of photos and/or other clusters, have a color and a name and are presented as colored strings of circles. **Workspaces** act as views on the underlying cluster/photo-model of the collection. One special workspace that cannot be manipulated by the user is the background, which is always visible, non-transformable and always shows the whole collection. The user is able to change the order (see below) of the background, but cannot translate photos or piles on it, so the order is always preserved (to prevent confusion as to whether an object lies at a certain position because of the order or because it was moved there).

Three techniques are used to maximize the users' overview: First, the background works as a fall-back point, where all photos are visible at any time in

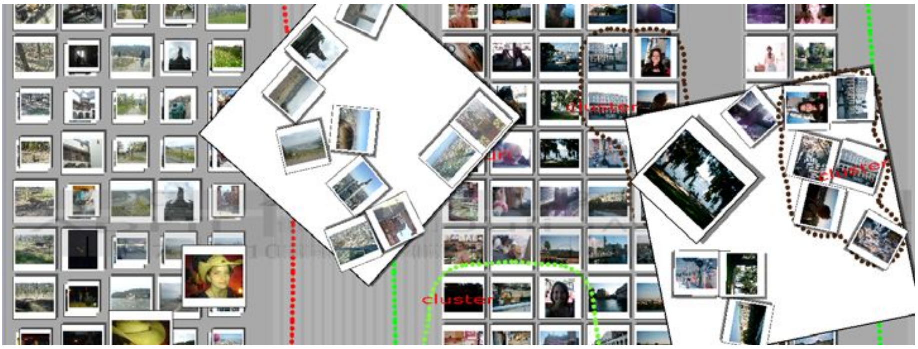


Fig. 1. A photo collection in Flux (time order, two additional workspaces)

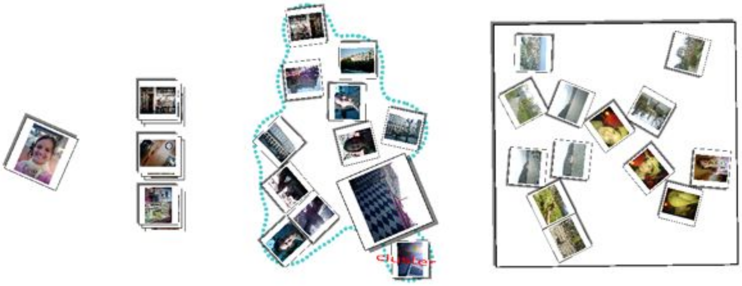


Fig. 2. Interface elements (from left to right): Photo, Piles, Cluster, Workspace

one given order and copies of them can be created and manipulated on additional workspaces. Second, to increase the number of photos visible, piles are automatically built thus reducing redundancy and visual clutter. Third, photos (on a workspace) and workspaces themselves behave like physical objects in the sense that they collide with one another and thus ensure that no object overlaps another and occludes information.

Four orders are available which can be changed independently for every workspace: Time, Similarity, Quality and Stream. **Time** shows all photos arranged along a calendar that is divided into a fixed number of intervals. The user can scale the time slots and make them either larger or smaller with a one- (moving one vertical border of the time slot) or two-finger gesture (moving both borders). The other time slots shrink or grow respectively. This approach allows a focusing on certain parts of the collection while preserving the complete global context (compared to, for example, a pure zoom-and-pan approach). Piles are built based on chronological adjacency and cluster membership. **Similarity** arranges all main-clusters vertically (and their subclusters horizontally) and builds piles out of visually similar photos. Within **Quality**, all photos are arranged based on their quality on a scale ranging from high quality on one side of the table to low quality on the other. The quality is symbolized by the color of the photos' borders that changes from green to red depending on the value. This quality value is automatically calculated based on the previously extracted low-level features and works with aspects such as blurriness, over-/underexposure, etc. For the sake of simplicity, the algorithm returns discrete values only, so a photo is either treated as "good" or as "bad". The last order, **Stream**, combines Time and Similarity: All clusters are again arranged vertically (subclusters lie close to their parents) with the photos shown along a horizontal axis based on the date they were taken (in this order there is no underlying (global) time line, so all photos are uniformly placed next to each other). The photos are scaled so that the contents of one whole cluster fit into the horizontal space. To optimize the number of photos displayed piles are built, but not purely based on time or low-level similarity but on the combination of the two: Every photo's similarity value is compared to its chronological neighbour's only, so that photos that were taken of the same motif or as snapshots in a row are automatically grouped and the chance that photos that are somehow related are combined into a pile is increased.

Interaction: Interaction in Flux happens on three different levels separated by dwell time (using additional graphical elements like buttons would have increased the on-screen clutter and forced the user to touch a certain section of the object). The user can choose the action's scope by touching the object and determine the type of action itself by waiting. The waiting time for a task depends on how frequently it is used: Transforming an object (translating, rotating, scaling) is readily available. Less common options like forming clusters become active after waiting for 500 miliseconds. Finally, rarely used options like changing the order of photos can be reached by waiting one second. A timeout is visualized by a change of color of the circle surrounding the user's finger or the appearance of a marking menu (depending on the type of action). Three geometric transformations are merged into two gestures: "Rotate'n'Translate" ([11]) rotates an object with a pseudo-physical friction while the user drags it with one finger. "Rotate'n'Scale" ([7]) lets the user set an object's scale with the distance of two fingers and its rotation with their movement. Except for changing appearances, these transformations can be used to delete photos and workspaces by scaling them below a certain threshold and unfolding piles by moving their topmost photo.

After waiting one timeout new clusters can be created by simply drawing a circle around the objects that should be contained and lifting the finger. When drawing a complete circle the circle's color switches and a lasso gesture becomes active: All objects that were selected (i.e., lay within the original circle) are copied to a new workspace, whose center lies at the position where the finger is finally lifted (both gestures are similar to the ones used in [8]).

Workspaces and clusters have marking menus that are shown after the second timeout: The workspace marking menu changes the current order. Cluster marking menus let the user adjust the color (directly with a color wheel) and the name (on an overlay writing field) or delete it. Choices are executed as soon as the user lifts the finger from the table.

Deployment: Flux was written in Microsoft C# within the .NET framework 2.0 and Direct3D 9. We relied on a framework by the University of Munich for low-level feature extraction and analysis, the PhysX™[5] physics engine by Ageia Technologies Inc. for the physical behavior of the objects and the Microsoft TabletPC SDK[6] for handwriting recognition in the naming of clusters.

The system was deployed on a horizontally mounted 42" LCD-screen with a touch-sensitive DViT-overlay[7] by Smart Technologies, which unfortunately limits the number of concurrent input points to two.

2.5 Discussion

Flux uses a combination of interaction and automation to ease the organization of a photo collection: The user gives a general structure by defining clusters and

[5] www.ageia.com/physx

[6] msdn2.microsoft.com/en-us/windowsvista/aa905027.aspx

[7] smarttech.com/DViT/

the enlarging and shrinking of parts of the collection in the Time-order, but the myriad of decisions of how to visualize the result (what objects to put where, what photos to use to build a pile) is left with the system. After presenting a visualization the user is still able to completely change it or force the system to repeat the process. Using this close coupling thus works better than an approach based purely on automation (that ignores the user's knowledge of the collection) or interaction (the leaves tedious tasks, e.g., checking the quality of a photo, with the user). Important in this regard is that all automatically performed actions are reversible (e.g., unfolding a pile to access photos) and that the underlying visualization mechanisms are graspable for the user by using animations. Still, limitations to this approach exist, especially when first starting Flux without a high-level organization by the user: The system is then forced to build piles of distinct photos that have to be tediously sorted into clusters. Here, the combination of the two backfires and even produces worse results than a standard scrollable or zoomable approach.

So, one point of improvement would be providing support for the initial organisation in a two-step fashion, where a pre-clustering can be quickly performed by the users which then leaves them with top-level clusters that provide an easier entry into the collection. Another option would be to automatically create initial clusters based on time ([3]). Additionally, large collections often become relatively cluttered especially on the background. A solution to this problem might be dropping the idea of complete overview at all times and allow the user to set the focus of his attention independently of the current order, for example, by allowing him to scale clusters on the background, thus reducing their recognisability but increasing the overall clarity.

3 Conclusion

In this paper we presented Flux, a photo organization tool with a focus on interaction and automation. By the close combination of those two a convenient workflow. For our current system, a formal evaluation is needed to gather if our design succeeded and to find overlooked weaknesses.

In a successor to Flux we might change the available orders - two of the four did not yield the expected results: Neither (binary) quality values nor one-dimensional similarity values should be visualized on a two-dimensional plane. The Similarity-order might be improved by changing the placement of clusters based on the average similarity of their members, so a visual Query-By-Example becomes possible. Still, it is doubtful whether a pure similarity view is useful at all ([4]) - a future version might merge the four orders into one with an emphasis on the chronological order ([10]) - possibly combining the Stream-order with the adaptability of time slots from Time and marking low-quality photos (e.g., displaying a small symbol on their border). Together with unlimited scaling of clusters on the background such a system would in all likelihood be more powerful and flexible than the current version and further elaborate on our concept of the coupling of interaction and automation.

References

1. Bederson, B.B.: Photomesa: a zoomable image browser using quantum treemaps and bubblemaps. In: UIST 2001: Proceedings of the 14th annual ACM symposium on User interface software and technology, pp. 71–80. ACM, New York (2001)
2. Huynh, D.F., Drucker, S.M., Baudisch, P., Wong, C.: Time quilt: scaling up zoomable photo browsers for large, unstructured photo collections. In: CHI 2005: CHI 2005 extended abstracts on Human factors in computing systems, pp. 1937–1940. ACM, New York (2005)
3. Cooper, M., Foote, J., Girgensohn, A., Wilcox, L.: Temporal event clustering for digital photo collections. ACM Trans. Multimedia Comput. Commun. Appl. 1(3), 269–288 (2005)
4. Rodden, K., Basalaj, W., Sinclair, D., Wood, K.: Does organisation by similarity assist image browsing? In: CHI 2001: Proceedings of the SIGCHI conference on Human factors in computing systems, pp. 190–197. ACM, New York (2001)
5. Shen, C., Lesh, N., Vernier, F.: Personal digital historian: story sharing around the table. Interactions 10(2), 15–22 (2003)
6. Hilliges, O., Baur, D., Butz, A.: Photohelix: Browsing, sorting and sharing digital photo collections. In: Second Annual IEEE International Workshop on Horizontal Interactive Human-Computer Systems(TABLETOP 2007), pp. 87–94. IEEE Computer Society, Los Alamitos (2007)
7. Apted, T., Kay, J., Quigley, A.: Tabletop sharing of digital photographs for the elderly. In: CHI 2006: Proceedings of the SIGCHI conference on Human Factors in computing systems, pp. 781–790. ACM, New York (2006)
8. Agarawala, A., Balakrishnan, R.: Keepin' it real: pushing the desktop metaphor with physics, piles and the pen. In: Proceedings of CHI 2006, pp. 1283–1292 (2006)
9. Kirk, D., Sellen, A., Rother, C., Wood, K.: Understanding photowork. In: CHI 2006: Proceedings of the SIGCHI conference on Human Factors in computing systems, pp. 761–770. ACM, New York (2006)
10. Rodden, K., Wood, K.R.: How do people manage their digital photographs? In: CHI 2003: Proceedings of the SIGCHI conference on Human factors in computing systems, pp. 409–416. ACM, New York (2003)
11. Kruger, R., Carpendale, S., Scott, S.D., Tang, A.: Fluid integration of rotation and translation. In: CHI 2005: Proceedings of the SIGCHI conference on Human factors in computing systems, pp. 601–610. ACM, New York (2005)

PhotoSim: Tightly Integrating Image Analysis into a Photo Browsing UI

Ya-Xi Chen and Andreas Butz

Media Informatics, University of Munich, Germany
yaxi.chen@ifi.lmu.de, andreas.butz@ifi.lmu.de

Abstract. Current photo browsers for personal photo collections mostly use the folder structure or camera-recorded time stamps as the only ordering principle. Some also allow manually provided meta-data (tags) for organizing and retrieving photos, but currently, only professional tools allow a pre-grouping of photos by image similarity. We believe that similarity is indeed a useful criterion both for image retrieval and casual browsing, and that the tight integration of content analysis techniques in media UIs in general can lead to more powerful UIs. In this paper we present a prototype, in which we have tightly integrated image analysis techniques and user feedback into a zoomable user interface for browsing and sorting digital photos. We have discussed our system with domain experts and received encouragement as well as valuable ideas for future research.

Keywords: Photo browsing, image analysis, personal photo collection, zoomable UI, clustering, user feedback.

1 Introduction

The digitalization of media influences many areas of our life, but often, the additional possibilities, which come with this change, are hardly used. Digital photography, for example, has brought along a steep increase in the overall number of photos taken, but the archives for digital photo collections mostly follow the same principles as those for analog photography. They use capture time and folder structures as the main ordering principles, and do not provide much advantage other than increased speed of retrieval [1]. In order to keep up with the growing amounts of data, novel paradigms for managing, archiving and retrieving digital photographs have become a major challenge for research, and these novel paradigms create the opportunity of more substantial changes in our way of dealing with digital media. According to studies [2], [4], users browsing their photo collections often do not have a specific search goal, or a technically rather unclear one, such as "look for a better photo of my daughter". Sometimes they might just browse for pleasure or story telling. At the same time, users are very reluctant to use tedious and time-consuming techniques, such as tagging each photo manually. Most of them prefer their photo collections to be organized automatically and hence cope with the organization by available metadata, such as folder hierarchy or camera-recorded capture time [1], [3]. Therefore, most current photo browsers are based on the latter.

A. Butz et al. (Eds.): SG 2008, LNCS 5166, pp. 224–231, 2008.

1.1 Related Work

The traditional UI for photo management (e.g., ACDSee, Picasa, iPhoto or Windows Explorer) uses a two-panel strategy. When the user selects a specific folder in the left panel, all photos in this folder are shown as thumbnails in the right panel, where the user can navigate within the folder by a vertical scrollbar and select thumbnails to see the original photo. More elaborate approaches include treemaps and zoomable UIs. In order to use the available screen space in an optimal way and provide a consistent mental model of the entire collection's organization, it has become a key research issue to maximize the usage of a given area and present as many photos as possible in it. One promising solution is a zoomable user interface, which presents information in one large plane and lets the user smoothly zoom in to see information in more detail, or zoom out to get an overview. Photomesa [6] is one of the popular zoomable photo browsers. It adopts a time-ordering and space-filling presentation strategy. Based on the theory of a quantum treemap [7], each directory is displayed in a different size depending on its content, which provides additional visual clues in the overview. All folders are ordered chronologically, and photos inside each folder are displayed in a non-hierarchical way. PhotoFinder [8], [9] aims at the organization of manually tagged photos. It lets the user organize photos into "collections" which are presented with a representative image and also enables the user to search in a boolean query interface. Other systems, such as TimeQuilt [11], PhotoToc [12] and CalendarBrowser [13], group photos into different events and offer automatic selection of a representative image. Even on large displays, researchers are looking for interactive and space-saving presentations, such as circular [14] and spiral [15] displays.

2 Tightly Integrating Similarity and User Feedback in the UI

Digital media collections allow us to derive additional ordering principles from the actual media content. In the case of photo collections, this means that image analysis can be used to derive low-level image features, which in turn can be used for the organization of images. This idea is used in a simple way in professional tools, such as Aperture [16], where successive pictures can be automatically grouped if they exhibit similar color histograms. This speeds up the initial phase of photo selection by conveniently partitioning the entire set into easily tractable subsets. Our goal is to go beyond this simple preselection mechanism by

- using more elaborate image features in the analysis process,
- applying this to entire collections or bigger subsets, not just to subsequent photos,
- tightly integrating image analysis in the UI for browsing and retrieving images,
- harnessing user feedback to improve over fully automatic techniques.

We are targeting a general audience, who produce high numbers of photos, but also are very reluctant to explicitly manually tag them. We believe that the tight integration of image analysis techniques in media UIs, combined with immediate user feedback, can partially make up for manual tagging and that it will lead to UIs, which operate close to human categorization principles, such as visual similarity. In our previous work [17], a fully automatic organization has been provided by low level

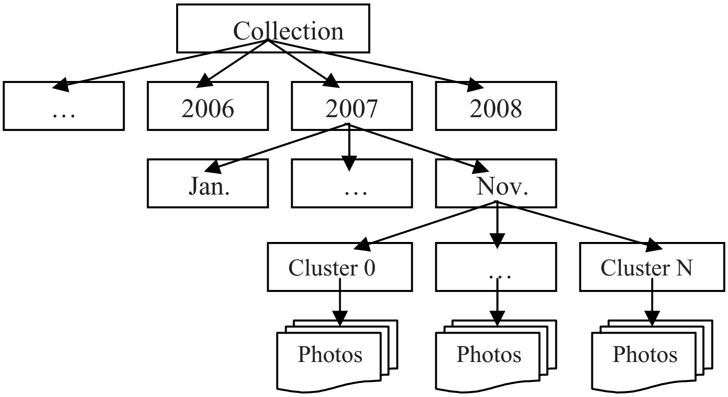


Fig. 1. Hierarchical organization of the photo collection according to capture time

image features. Our current attempt to create such a UI enriched by content analysis is a zoomable photo browser, which uses image analysis to further structure photo collections below the level of time, but allows manual overrides of this ordering and infers feedback from the manipulations used for overriding. On a general level, the collection is organized hierarchically according to capture time (see Figure 1), and forms a tree with subsets, where photos can be grouped further according to their similarity. In order to do this, we compute low level image features in a preprocessing step and then cluster the images according to these features.

3 The PhotoSim User Interface

Our photo browsing application was implemented based on the prefuse toolkit for interactive Information Visualization [19]. PhotoSim presents photos clustered by time and content similarity and provides basic interaction techniques, such as drag-and-drop, pan and zoom. User feedback is derived implicitly from mouse operations (see Integration of user feedback).

The UI contains four main panels: a graph view, a tree view, an overview and a control panel, as shown in Figure 2. The three view panels act as coordinated multiple views. In the tree view, when a day, month, year or the entire collection is selected, all the corresponding photos appear in the graph view, creating a potentially very large graph. Photos can be clustered in this way at different levels of the temporal hierarchy. This allows, for example, finding similar photos across different days, months or even years.

The graph view is not limited in size, thereby not limiting the overall number of photos and clusters which can be shown. The user can freely pan and zoom within the graph view, but will always be provided with an overview of the entire graph in the overview pane in the top right corner. In the control panel (bottom right), the user can adjust the threshold used for clustering and explicitly save the current arrangement.

Since the location of each photo relative to the cluster center is determined by its similarity to the centroid photo and neighboring photos, a dent in the cluster shape

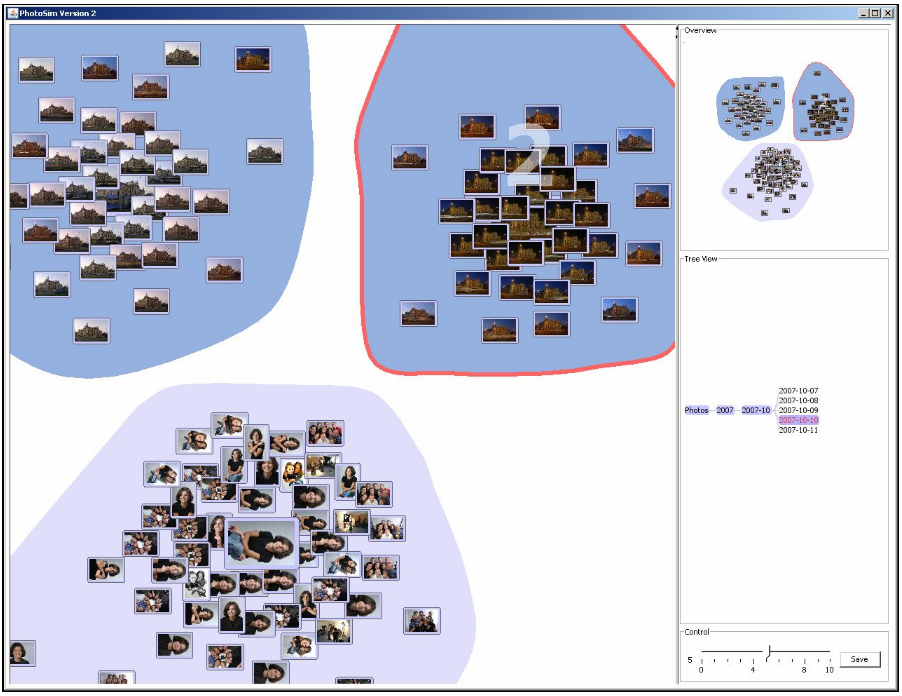


Fig. 2. The user interface of PhotoSim contains a graph view on the left and a column with an overview pane, a tree view, and a control panel on the right

visually identifies outliers, which are less similar than other photos in the cluster. These outliers can easily be dragged to other clusters according to the user's better judgment of similarity, or to any other ordering principle he/she may think of.

4 Image Analysis

The image analysis we use, entails several processing steps. First, a set of low level features is extracted from the photos. Based on these features, photos are then grouped into different clusters.

4.1 Low Level Features Used

In order to compute similarity between photos, we use color features, textural features and roughness. Color features are computed in YUV color space, which has proven to yield results which are more consistent with human perception [17]. The U and V components are partitioned into 6 bins, which lead to a color histogram with 36 dimensions. Four related statistic features are also extracted: mean, standard deviation, skewness and kurtosis of the color moment in YUV [21], yielding an overall of 48 color-related features. Besides these, we extract Haralick features [20], which provide easily computable textural features based on gray-tone spatial dependencies and finally also roughness statistics derived from the gray level variation. In total, 161 features are extracted. Feature extraction does not work in real time and is done in a preprocessing step. The preprocessing of a collection of 813 photos (2M Pixel each) took about 2 hours in our current implementation.

4.2 Clustering Algorithm

After extracting these low level features, the photos are grouped using a standard clustering algorithm. Since we cannot make any well-grounded assumptions about the photo content in the general case, the number of clusters in particular is unknown beforehand. Since this would have to be known for a supervised clustering process, we can only employ an unsupervised clustering algorithm. In our implementation, we use Simple K-Means [22], [23]. Figure 2 shows the clustering result of a photo collection with 3 different motives: portrait, building in daylight and at night. The clustering algorithm correctly groups them into 3 clusters, which is consistent with human perception in this case. Depending on the actual degree of similarity between and within motives, this approach might still create too many or too few clusters in the general case. Although this might also be tweaked (see Future Work), we are convinced that a fully automatic approach will never work perfectly in all cases.

4.3 Integration of User Feedback

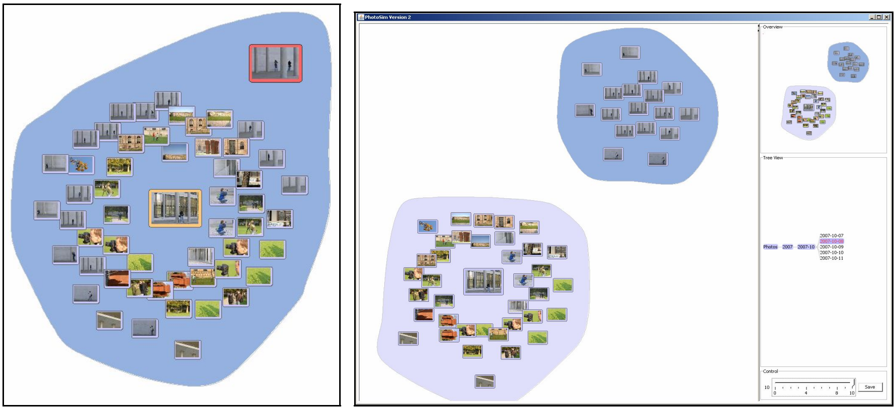
From the discussion above, it becomes clear that it will never be possible to perfectly match a user's visual understanding by a fully automatic approach. One solution to this problem is to take the human into the loop and combine the image analysis techniques with user feedback. We therefore provide manual overrides of the automatic results. The simplest such override is to just drag a photo from its original cluster into another existing cluster. On top of this, there are two ways to create new clusters:

4.3.1 Create a New Cluster Manually

First, the user can set the slider for the clustering threshold to zero, which means he/she wants to create cluster manually. Then the user can drag a photo outside its original cluster, and a new cluster will be created which contains this photo. If the user wants to add more photos in this newly created cluster, he/she can then manually drag other photos into this cluster one by one. This functionality can be used for permanent storage of personal favorites, or to create clusters as temporary structures for storytelling. Since all these operations are executed by the user manually, they are regarded as intentional and saved permanently.

4.3.2 Create a New Cluster Automatically

The user can also automatically create a new cluster of similar images from a manually selected example photo. First, he/she has to adjust the slider in the control panel to select a clustering threshold other than Zero. Then the user can drag the example photo outside the original cluster, as Figure 3(a) shows. A new cluster will be created which contains this example photo and other similar photos in the currently selected time period, as Figure 3(b) shows. If the clustering threshold is set to a high value, this means that the required degree of similarity is relative low, and therefore some



(a) (b)

Fig. 3. Creating a new cluster automatically: (a) Dragging the example photo outside the original cluster. (b) The newly created cluster contains the example photo and other similar photos.

other less similar photos will be also included in the new cluster. Since the result of this functionality is somewhat less predictable, the new cluster is not saved automatically in order not to destroy the existing presentation. If the user is satisfied with the result, he/she can click the "Save" button in the control panel to save it permanently.

5 Preliminary Evaluation

We have tested our prototype with five different photo collections. Four of them result from a photography workshop and were taken in the course of three days by four different ambitioned amateur photographers. The size of the collections varies between 317 and 812. Another collection was offered by a former professional photographer and contained a selection of 603 photos in total, captured over five years.

Although we have not conducted a formal user study, the discussions with the authors of our example photo collections were quite encouraging. The former professional photographer said that he would like to use such a photo browser even more, if the structure of the collection after clustering and manual modifications could be translated back into a directory structure in the file system. He claimed that this might be a very valuable tool for potentially reorganizing his entire photo collection, not just the selection used in the test.

6 Summary and Future Work

In this paper, we have described an example for the tight integration of content analysis in a media UI and we have shown how this can lead to a more powerful set of manipulations for browsing and sorting of the media content. We have also seen that the tight integration with the UI is important, because it allows the user to manually override the automatic decisions of the system by providing implicit feedback. Although several forms of feedback are provided in our prototype, we still think that the interaction could be improved by allowing the user to organize photos in a more natural way, e.g., by supporting a more flexible and fluent sorting functionality. One parameter, which probably merits even more attention, is the average size and overall number of clusters for any given subset of a collection. The correct value will largely depend on personal taste and the actual content of the collection. Finally, if the similarity between any pair of two media items in the entire visualization could be visually approximated by the distance between them, this might close the gap towards self-organizing media networks as described in [25].

Acknowledgments

This research was funded by the Chinese Scholarship Council (CSC) and by the German state of Bavaria. We would like to thank the photographers who provided their collections as test sets and their opinions about our prototype, and we are also very grateful to Dominikus Baur, Michael Sedlmair and Marc Breisinger for their valuable input to this work.

References

1. Rodden, K., Wood, K.: How do people manage their digital photographs? In: CHI 2003: Proceedings of the SIGCHI conference on Human factors in computing systems, Ft. Lauderdale, Florida, USA, pp. 409–416. ACM Press, New York (2003)
2. Drucker, S., Wong, C., Roseway, A., Glenner, S., De Mar, S.: Mediabrowser: reclaiming the shoebox. In: AVI 2004: Proceedings of the working conference on Advanced visual interfaces, Gallipoli, Italy, pp. 433–436. ACM Press, New York (2004)
3. Frohlich, D., Kuchinsky, A., Pering, C., Don, A., Ariss, S.: Requirements for photoware. In: CSCW 2002: Proceedings of the 2002 ACM conference on Computer supported cooperative work, pp. 166–175. ACM Press, New York (2002)
4. Kirk, D., Sellen, A., Rother, C., Wood, K.: Understanding photowork. In: CHI 2006: Proceedings of the SIGCHI conference on Human Factors in computing systems, Montréal, Québec, Canada, pp. 761–770. ACM Press, New York (2006)
5. Spence, R.: Information Visualization: Design for Interaction, 2nd edn. Prentice Hall, Englewood Cliffs (2006)
6. Bederson, B.B.: PhotoMesa: a zoomable image browser using quantum treemaps and bubblemaps. In: UIST 2001: Proceedings of the 14th annual ACM Symposium on User interface Software and Technology, pp. 71–80. ACM Press, New York (2001)
7. Shneiderman, B.: Tree visualization with treemaps: a 2-D space-filling approach. ACM Transactions on Graphics 11, 92–99 (1992)
8. Kang, H., Shneiderman, B.: Visualization methods for personal photo collections: browsing and searching in the photoFinder. In: ICME 2000: Proceedings of IEEE International Conference on Multimedia and Expo., New York, NY, USA, pp. 1539–1542 (2000)
9. Shneiderman, B., Kang, H.: Direct annotation: a drag-and-drop strategy for labeling photos. In: IV 2000: Proceedings of IEEE International Conference on Information Visualization, London, England, pp. 88–98 (2000)

10. North, C., Shneiderman, B., Plaisant, C.: User controlled Overviews of an image library: A Case Study of the Visible Human. In: Proceedings of the first ACM international conference on Digital libraries, Bethesda, Maryland, USA, pp. 74–82 (1996)
11. Huynh, D., Drucker, S., Baudisch, P., Wong, C.: Time quilt: Scaling up zomable photo browsers for large, unstructured photo collections. In: CHI 2005: extended abstracts on Human factors in computing systems, pp. 1937–1940 (2005)
12. Platt, J., Czerwinski, M., Field, B.: Phototoc: automatic clustering for browsing personal photographs. Microsoft Research Technical Report MSR-TR-2002-17 (2002)
13. Graham, A., Garcia-Molina, H., Paepcke, A., Winograd, T.: Time as essence for photo browsing through personal digital libraries. In: Proceedings of the 2nd ACM/IEEE-CS joint conference on Digital libraries, Portland, Oregon, USA, pp. 326–335. ACM Press, New York (2002)
14. Shen, C., Lesh, N.B., Vernier, F., Forlines, C., Frost, J.: Sharing and building digital group histories. In: CSCW 2002: Proceedings of the 2002 ACM conference on Computer supported cooperative work, New Orleans, Louisiana, USA, pp. 324–333. ACM Press, New York (2002)
15. Hilliges, O., Baur, D., Butz, A.: Photohelix: browsing, sorting and sharing digital photo collections. In: TABLETOP 2007: Second Annual IEEE International Workshop on Horizontal Interactive Human-Computer Systems, Newport, RI, USA, pp. 87–94 (2007)
16. Aperture, software of phto editing and management, http://www.apple.com/aperture
17. Hilliges, O., Kunath, P., Pryakhin, A., Kriegel, H., Butz, A.: Browsing and sorting digital pictures using automatic image classification and quality analysis. In: Proceedings of HCI International 2007, Beijing, China, pp. 882–891. Springer, Heidelberg (2007)
18. Smeaton, A.F.: Content vs. context for multimedia semantics: the case of SenseCam image structuring. In: Avrithis, Y., Kompatsiaris, Y., Staab, S., O'Connor, N.E. (eds.) SAMT 2006. LNCS, vol. 4306, pp. 1–10. Springer, Heidelberg (2006)
19. Prefuse, interactive information sisualization toolkit, http://www.prefuse.org/
20. Haralick, R.M., Dinstein, I., Shanmugam, K.: Textural features for image classification. IEEE Transactions on Systems, Man, and Cybernetics, 610–621 (1973)
21. Stricker, M., Orengo, M.: Similarity of color images. In: SPIE 1995: Proceedings of Storage and Retrieval for Image and Video Databases, San Jose, CA, pp. 381–392 (1995)
22. Kanungo, T., Mount, D.M., Netanyahu, N.S., Piatko, C.D., Silverman, R., Wu, A.Y.: Analysis of a simple k-means clustering algorithm. In: 16th Annual Symposium on Computational Geometry, pp. 100–109 (2000)
23. Weka, data mining with open source machine learning software, http://www.cs.waikato.ac.nz/~ml/weka/
24. McQueen, J.: Some methods for classification and analysis of multivariate observations. In: 5th Berkeley Symposium on Mathematics, Statistics, and Probabilistics, pp. 281–297 (1967)
25. Nürnberger, A., Detyniecki, M.: Weighted Self-Organizing Maps: Incorporating User Feedback. In: Kaynak, O., Alpaydın, E., Oja, E., Xu, L. (eds.) ICANN 2003 and ICONIP 2003. LNCS, vol. 2714, pp. 176–184. Springer, Heidelberg (2003)

Thor: Sketch-Based 3D Modeling by Skeletons

Romain Arcila[1], Florian Levet[2], and Christophe Schlick[1]

[1] Iparla Project Team (INRIA Bordeaux Sud-Ouest, Bordeaux University)
romain.arcila@etu.u-bordeaux1.fr, schlick@labri.fr
[2] PICIN, Magendie institute, Inserm U862, Bordeaux University
florian.levet@inserm.fr

Abstract. The creation of freeform 3D objects has been historically done with a 3D modeler, either based on CSG operators, spline surfaces or subdivision surfaces. However sketch-based modeling, initiated by the groundbreaking Teddy system [1], has recently emerged as a more intuitive alternative. Following Teddy, sketch-based modeling is usually done by letting the user sketch the silhouette of the generated mesh. In this paper, we present a 3D sketching prototype based on the skeleton as input of the system.

1 Introduction

In order to create user interfaces simpler than classical 3D modelers and to enable new users to model interesting 3D objects, alternative approaches have recently emerged, based on the human ability to quickly draw a global overview of an object. These approaches are called *3D Sketching*. Their principle is to easily infer a 3D shape from a sketch and add details thanks to different editing operations (e.g., cutting, extrusion), all based on 2D gestures and curves.

Teddy [1] is the precursor of 3D freeform modeling tools based on this paradigm. From a sketched 2D silhouette, a skeleton is extracted and the 3D mesh of the object is inferred based on a circular profile-curve. Moreover, a gesture grammar converts drawn curves into corresponding modeling operations: extrusion, deformation, cutting, etc. Since the interface is totally based on sketches, even users that are non-expert in 3D modeling software can create interesting objects. Both the interaction gesture and the geometric models have been improved by following works. For instance, implicit [2,3,4] and volumetric [5] surfaces have been used to smooth and straighten the geometric representation of the objects. Several systems [3,6,7] have used profile-based editing, based on the idea that global modifications of an object are more easily done by changing the overall profile-curve applied on the model. Moreover, with freeform profiles, users are not limited to blobby objects. Finally, recent approaches have related the power of subdivision surfaces to 3D sketching systems [8] or developed a system using multiple views [9] to simplify the modeling process.

All these techniques are based on the Teddy paradigm which is to create 3D models starting from a 2D silhouette. But, in some cases, it may be more interesting and easy to create an object from its skeleton. There are several classical 3D modeling techniques that are based on skeletons, the most famous is

A. Butz et al. (Eds.): SG 2008, LNCS 5166, pp. 232–238, 2008.

undoubtedly the generalized cylinder. In [10], Grimm and Hughes presented such a system, where the idea was to generate a sweep surface starting from a skeleton sketch and then, to interactively modify its profile to edit the shape of the objects. The two main limitations of this system is the use of an implicit definition for the objects, which limits the variety of generated shapes and the impossibility to create objects with branches. Some other sketching systems [11,12] are also using generalized cylinders as part of their modeling process. For instance, Ijiri et al. [12] used them as the way to model the stem of flowers, while Kim and Neumann [11] used them to create clusters of hair. Here again, the main limitation is the impossibility to create objects with branches.

In this paper, we present *Thor*, a prototype system that uses sketched skeletons to generate 3D models. The creation process implemented in Thor is quite similar to the one developed by Ijiri et al. [12]. The idea is to first sketch the main skeleton of the object and then to add branches in order to create more detailed and complicated objects (see Section 2). Since one of our main concern is the possibility to port our system to mobile devices (with limited computing capacities and limited screen sizes), subdivision surfaces are a perfect choice for the geometric representation of the underlying 3D models since their complexity can easily be adapted to the resources available from the device. Section 3 gives a quick overview of the interaction process designed in Thor to create a 3D model. Moreover, to simplify user interaction on mobile devices, we also propose the *CrossWidget*, a widget that replaces the standard menubar and toolbar, by a simplified cross-shaped popup menu (see Section 3.2).

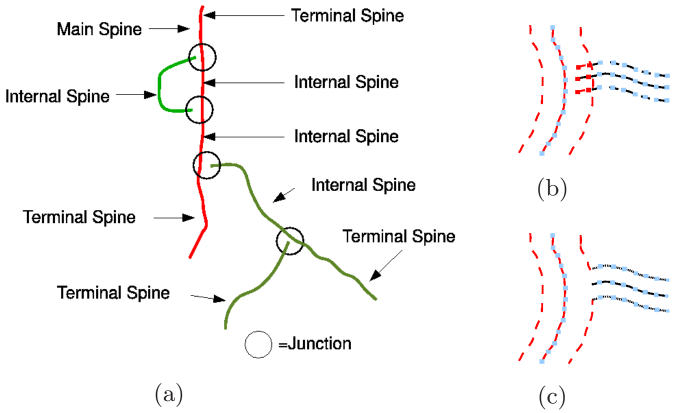
2 Generation of the Object

2.1 Construction of the Skeleton

The system takes as input a skeleton, defined by a set of curves. This skeleton goes through an analysis system that classifies each topological subset into one of the four categories: junction point, main spine, internal spine or terminal spine (see Fig.1(a)).

Each curve composing the skeleton is filtered and uniformly resampled using the arc-length parametrization of a Catmull-Rom spline. This process efficiently removes high frequency noise often present in hand-draw sketches. The first curve drawn by the user is defined as the *main spine*. Each time a new skeleton curve is added, its junction point to the actual skeleton has to be determined. First, the starting point of the new skeleton curve has to be close enough to an existing spine (in the other case, the curve is discarded). Then, since each curve has been resampled as a set of skeleton points, the new skeleton curve is connected to the closest skeleton point laying on an existing spine. This point is then classified as a junction point. (Note that a curve may have both its starting and ending points classified as junction points as can be seen in Fig.1(a)).

Once the junction points have been determined, the classification of the skeleton spines is quite straightforward: a spine located between a junction point and



(a)

(b)

(c)

Fig. 1. The two steps of the skeleton construction: (a) spine classification, (b) and (c) deletion of unwanted points at spine junction

an ending point is identified as a terminal spine and a spine between two junction points is classified as an internal spine (see Fig.1(a)).

Now that we have classified all spines composing the skeleton, Thor sets their thickness by letting the user draw a radius for each spine. Finally, for each junction, when the thickened spines overlap each other, all the points of the branching spine that are inside the thickness of the original one are removed by the system (see Fig.1(b) and 1(c)). This provides a clean object where all parts are clearly separated and simplifies the construction of the mesh.

2.2 Construction of the Mesh

Once the skeleton has been defined, the next step consists in generating the corresponding mesh for the object. Here again, the situation is straightforward when there is only one a single spine: if the spine has only one point, we build a sphere, in the other case, we build a generalized cylinder with two caps.

For more complex skeletons, we build a generalized cylinder with one cap (resp. no caps) for each terminal (resp. internal) spine. By default, the section used for the extrusion is a square. This allows easy junction between branching spines by connecting quad meshes. The result of this step is one global base mesh for the whole object, exclusively composed of quads. This base mesh can then be easily subdivided to generate a smooth and blobby shape, by using the standard Catmull-Clark subdivision process.

In order to get non-blobby object, edges and vertices may be tagged as sharp. Therefore, when the mesh is subdivided, these edges and vertices will not be smoothed as explained in [13].

3 Interaction

In its current implementation, our prototype presents two different views of the object under construction (see Fig.2 for a snapshot of our system). All

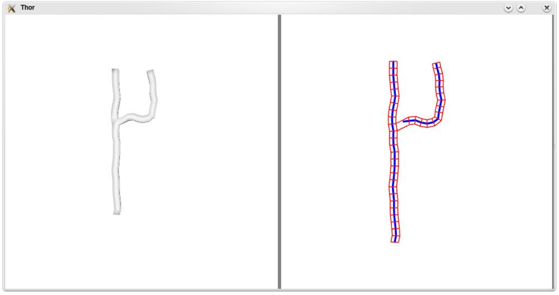


Fig. 2. The 2 views of our interface. Left is the 3D View, which displays the objects composing the scene, right is the Skeleton Editor which displays the uniformly sampled skeleton of the selected object.

interactions performed on the objects are done in the *3D View* while the *Skeleton Editor* displays the uniformly sampled skeleton and the silhouette of the selected object and allows users to modify them.

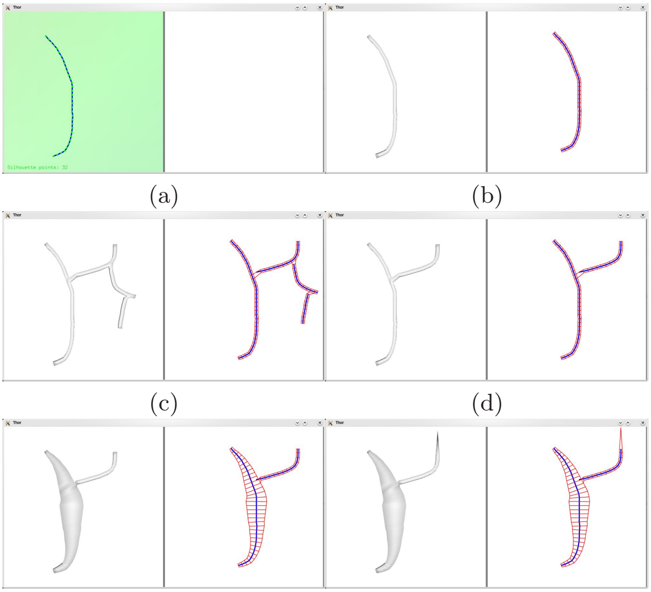
3.1 Interactive Creation of a 3D Model

In order to create objects, our system needs a sketch of the skeleton drawn in the 3D View. After drawing the main spine (see Fig.3(a) and(b)), the user may add as many spines as he wants as soon as their starting point is close enough to the main spine. When the skeleton drawing is complete, a simple line intersecting one of the spines launch the inflating process, with the object thickness being dependent of the length of the intersecting curve. Then, the 3D object is displayed in the 3D view while its skeleton and silhouette are displayed in the Skeleton Editor. Starting from now, it is not possible anymore to add spines on the object in the 3D View.

In order to modify the object shape, users have access to a set of different actions, all performed on the Skeleton Editor. For instance, users can add new spines on an object (the process being similar to the one performed during the creation process in the 3D View) (see Fig.3(c)) or remove ones (see Fig.3(d)). Please note that all the spines dependent of this deleted one (spines that were created after this spine and connected to it) are discarded too (see Fig.3(d)).

After selecting a spine region, users can change its thickness. The system then interpolates the overall spine thickness with respect to the precedent one (defined at the non-selected region of the spine) and the newly one defined (see Fig.3(e)).

As said in Section 2, the default creation process for the extremities of the base mesh of an object is a square (see Fig.3(e)) which leads, when subdivided, to a blobby 3D model. In order to create models with straight edges, two actions are designed in our system. First, it is possible to point extremities of a spine (see Fig.3(f)). Moreover, it is possible to mark an edge as a sharp one. Thus, when subdivided, the 3D model keeps the straight edges as can be seen in Fig.5(b) (please note that it is possible to mark a mesh edge as sharp in the 3D view).



(a) (b)

(c) (d)

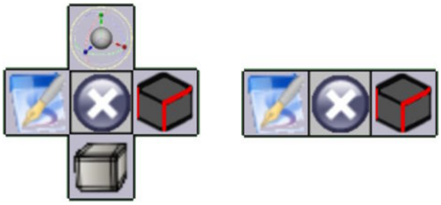
(e) (f)

Fig. 3. Interactive creation of an object. (a) Drawing the main spine. (b) Creation of the mesh based on the main spine. (c) Adding spines to the skeleton. (d) Removal of the middle spine, please note that the spine tied to the deleted one have been automatically removed. (e) Changing the thickness of the main spine. (f) The extremity of the right spine is set as pointed.

3.2 The CrossWidget

As said in the introduction, we mainly focus our 3D sketching system for mobile devices (i.e. with small screen sizes and without keyboard). Thus, to remove the need of keyboard, menus and toolbars, we have developed a new input widget, called *CrossWidget* (see Fig.4).

The CrossWidget is a restricted form of PieMenu [14]. The main advantage of PieMenu compared to standard linear menu is that the popup is centered at the current cursor location, which reduces the movement required by the



(a) (b)

Fig. 4. The CrossWidget with (a) all action defined and (b) only three actions defined

item selection and thus reduces the user fatigue. PieMenu was first designed to simplify user interaction by having direct access to a limited number of the available commands. It is particularly interesting for new users, as it offers a gentle learning curve. Unfortunately, many implementation of the PieMenu add more and more commands to it. In Maya, PieMenus are now completely overcrowded and not usable for non-experts. To avoid this extra complexity, our CrossWidget is limited by definition to only five actions.

The CrossWidget is an input widget displayed as a cross composed of five squares, which can handle up to five actions: each square represents a simple action, defined by the application and eventually modified by the user. If one of the square has no corresponding action defined, the default behavior of the CrossWidget is to hide the corresponding square.

In our current implementation, three CrossWidgets are defined. Each action of each CrossWidget has its own way to interpret a sketch. Two CrossWidgets are defined in the 3D View: the first presents the following actions: create a new skeleton, swap shader (change the render type), subdivide the selected object, mark edge as sharp on the object, and swap to the second CrossWidget, i.e. transform CrossWidget. The transform CrossWidget allows the following actions: translate, scale, rotate and switch to the standard CrossWidget. The transform CrossWidget is only necessary on devices with no keyboard (since, in classical systems, the 3D interaction of objects is performed by a combination of keyboard keys and mouse). The Skeleton Editor's CrossWidget has the following actions: add spines, remove spines, change thickness of a part of a spine, mark/unmark sharp and oversketch spines.

4 Conclusion and Future Work

In this paper we have presented Thor, a system that allows interactive creation and edition of objects based on their skeleton, as can be seen in Fig.5 where two objects created with our prototype are shown. For user convenience, all modifications performed on the Skeleton Editor are interactively and immediately reported on the generated mesh. The generated mesh are smoothed by the use of subdivision surface, however the system provides a way to mark edges as sharp and thereby to have non-smooth shapes.

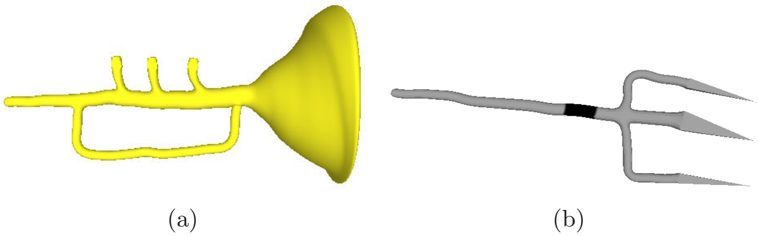


Fig. 5. Two objects created in few seconds with our system: (a) a trumpet, (b) a trident. Please note that the spikes of the trident are not smoothed even after being subdivided.

Since our current system is only a prototype, a lot of promising research directions exist. First, we are planning to extend our system in order to allow greater interaction. The first direction is to improve the skeleton generator himself by adding other features such as non-planar skeleton which would help creating more complex objects such as trees. Moreover, we plan to develop a skeleton generation based on curvature in order to limit the small variations currently noticeable on the reconstructed 3D models. Other planned features are the possibility to interactively modify the section curve used for the extrusion and the possibility to define loops in the skeleton. The second direction is to provide an hybrid system, in which a shape can be specified either by its silhouette or by its skeleton.

References

1. Igarashi, T., Matsuoka, S., Tanaka, H.: Teddy: a sketching interface for 3D freeform design. In: Proc. of ACM SIGGRAPH 1999 (1999)
2. Karpenko, O., Hughes, J., Raskar, R.: Free-form Sketching with Variational Implicit Surfaces. In: Proc. of Eurographics 2002 (2002)
3. Tai, C.L., Zhang, H., Fong, C.K.: Prototype Modeling from Sketched Silhouettes based on Convolution Surfaces. Computer Graphics Forum 23(1) (2004)
4. Schmidt, R., Wyvill, B., Sousa, M., Jorge, J.: ShapeShop: Sketch-Based Solid Modeling with BlobTrees. In: Eurographics Workshop on Sketch-Based Interfaces (2005)
5. Owada, S., Nielsen, F., Nakazawa, K., Igarashi, T.: A Sketching Interface for Modeling the Internal Structures of 3D Shapes. In: Butz, A., Krüger, A., Olivier, P. (eds.) SG 2003. LNCS, vol. 2733. Springer, Heidelberg (2003)
6. Cherlin, J.J., Samavati, F., Sousa, M.C., Jorge, J.A.: Sketch-based modeling with few strokes. In: SCCG 2005: Proc. of the 21st spring conference on Computer graphics, pp. 137–145 (2005)
7. Levet, F., Granier, X.: Improved Skeleton Extraction and Surface Generation for Sketch-based Modeling. In: GI (Graphics Interface) (2007)
8. Wang, H., Markosian, L.: Free-form sketch. In: Eurographics Workshop on Sketch-Based Interfaces (2007)
9. Levet, F., Granier, X., Schlick, C.: Multi-View Sketch-based FreeForm Modeling. In: Butz, A., Fisher, B., Krüger, A., Olivier, P., Owada, S. (eds.) SG 2007. LNCS, vol. 4569. Springer, Heidelberg (2007)
10. Grimm, C., Hughes, J.: Implicit generalized cylinders using profile curves. In: Implicit Surfaces, pp. 33–41 (June 1998); Creating sweep surfaces using sketching
11. Kim, T.Y., Neumann, U.: Interactive multiresolution hair modeling and editing. In: SIGGRAPH 2002: Proceedings of the 29th annual conference on Computer graphics and interactive techniques, pp. 620–629. ACM, New York (2002)
12. Ijiri, T., Owada, S., Igarashi, T.: Seamless integration of initial sketching and subsequent detail editing in flower modeling. Computer Graphics Forum 25(3), 617–624 (2006)
13. Biermann, H., Levin, A., Zorin, D.: Piecewise smooth subdivision surfaces with normal control. In: SIGGRAPH 2000: Proceedings of the 27th annual conference on Computer graphics and interactive techniques, New York, NY, USA, pp. 113–120. ACM Press/Addison-Wesley Publishing Co. (2000)
14. Callahan, J., Hopkins, D., Weiser, M., Shneiderman, B.: An empirical comparison of pie vs. linear menus. In: CHI 1988: Proceedings of the SIGCHI conference on Human factors in computing systems, pp. 95–100. ACM, New York (1988)

Sketch-Based Navigation in 3D Virtual Environments

Benjamin Hagedorn and Jürgen Döllner

Hasso-Plattner-Institut at University Potsdam,
Prof.-Dr.-Helmert-Str. 2-3, 14482 Potsdam, Germany
`benjamin.hagedorn, doellner@hpi.uni-potsdam.de`

Abstract. Navigation represents the fundamental interaction technique in 3D virtual environments (3D VEs) as it enables the users to explore the 3D world and to interact with its objects. Efficient navigation strategies and techniques are required, which take account of the users and their goals and avoid problems of general navigation methods, such as "getting-lost" situations and confusing view configurations. This paper presents a novel method for specifying and controlling navigation in 3D VEs based on sketching navigation commands. The users sketch their navigation intentions on top of the perspective projection of the 3D scene. The system interprets these sketches regarding their geometry, spatial context, and temporal context. Unlike other sketchy navigation techniques, our approach identifies the hit objects of the underlying 3D scene and takes advantage of their semantics and inherent navigation affordances. The approach has been prototypically implemented for the exploration of a virtual 3D city model with a touch-sensitive display.

1 Introduction

Navigation, which is often referred to as "the aggregate task of wayfinding and motion" [4], denotes the fundamental interaction technique in 3D geovirtual environments (3D VEs) but still represents a non-trivial task for the user interface technology [7]. General navigation techniques (e.g., world-in-hand controls, fly-over controls, and virtual trackballs) give the users direct control over the navigation process. Common problems of these approaches include "getting-lost" situations, confusing view configurations, abrupt camera motion, and loss of visual contact to landmarks. Thus, to improve the usability of 3D interaction processes, navigation techniques have to assist users to navigate through and interact with the 3D world and its objects.

This paper presents a sketch-based navigation technique for 3D VEs such as virtual 3D city models and 3D landscape models. In our approach, the users express their navigation intentions in terms of graphical sketches by drawing curves and points to indicate paths and locations as well as pen-based gestures. These sketches are interpreted according to a pre-defined graphical vocabulary of navigation commands, taking into account their spatial and temporal context as well as the navigation affordances inherent to the elements of the 3D VE.

A. Butz et al. (Eds.): SG 2008, LNCS 5166, pp. 239–246, 2008.

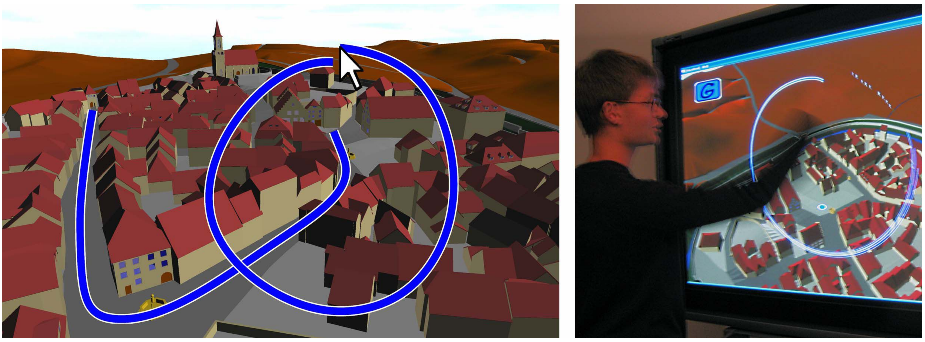


Fig. 1. Left: Example of a sketch-based navigation command applied to a complex virtual 3D city model. The user draws a curve on the street and combines this command with a circle like gesture. The derived animation will move the camera along the sketched path and finally rotate for inspecting the target area. – Right: Using the sketch-based navigation with a touch-sensitive smart board.

Sketch-based navigation is conceptually a higher-level navigation technique, as it relieves the users from controlling the motion task and even can assist in wayfinding. It can be used with any device allowing for 2D sketch input, e.g., mouse, graphic tablet, or touch-screen – see Fig. 1.

2 Related Work

Navigation constraints cope with the large number of degrees of freedom inherent to 3D navigation and represent a major technique in the field of assisted 3D navigation. Hanson and Wernert [9] propose designer-supplied constraints on the bases of 2D controllers, Tan et al. [13] introduce the speed-coupled flying, and Buchholz et al. [1] apply several navigation strategies for reaching a high orientation value. A detailed description of constraints for navigation in geovirtual environments can be found at Döllner [5]. StyleCam by Burtnyk et al. [2] represents an authoring-based approach constraining the camera movements in space and time. HoverCam by Khan et al. [12] assists users in panning, zooming and tumbling the virtual camera, particularly for single object inspection. Different from those, Russo dos Santos et al. [7] propose a navigation concept, which provides specific navigation methods according to the type of a 3D VE.

Sketching is a natural and intuitive input technique, which can be applied for 2D and 3D interaction. In the context of virtual 3D objects and virtual 3D worlds, sketching is often regarded as an effective means for modeling and manipulation [3][11][14]. Only few approaches seem to target at sketch-based navigation in 3D VEs. Igarashi [10] et al. introduce the concept of path drawing for 3D walkthroughs, which allows the users to sketch the path a virtual avatar shall move along on top of the perspective view of the 3D scene. Cohen et al. [3] suggest a method for sketching 3D curves and mention the possibility to use them as a camera path and Hachet et al. [8] propose a technique, which uses a circle-shaped gesture for focus definition and a special widget for positioning the camera. Döllner et al. [6] present an approach for semantics-based navigation in 3D city models on mobile devices, which bases on sketching navigation commands by stylus and incorporates the type and meaning of the 3D city objects for deriving an appropriate navigation animation. The article at hand gives a detailed description of an extended sketch-based navigation concept and implementation.

3 Sketch-Based Navigation

3.1 Sketchy Navigation Commands

As described in [6], we conceptually distinguish two types of navigation sketches:

- *Object-related sketches* are associated with objects of the 3D VE and enable a semantics-based interpretation as the system can derive the intended navigation from the semantics of the marked scene objects.
- *Pen-based gestures* implement a complementary set of commands that are not bound to a spatial context.

Sketchy navigation commands allow for the following types of navigation interactions, which differ in their degree of navigation abstraction:

- *Camera-oriented navigation.* Users perform wayfinding by themselves. They think in terms of moving the camera "left and right", "up and down", etc.
- *Motion-oriented navigation.* Users sketch by drawings, where the camera shall move along and where to gaze. This allows for the definition of more complex navigation patterns, such as "drive along this path and look at that building".
- *Task-oriented navigation.* Users sketch complete tasks or subtasks to be fulfilled, e.g., "get an overview" or "inspect the building". Particularly, they no longer deal with camera positions and orientations or camera paths and gaze directions. Of course, task-oriented navigation commands heavily depend on the users and the user tasks.

Table 1. Examples of pen-based gestures. The black dot indicates a gesture's starting point.

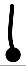
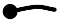
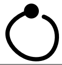
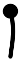
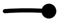
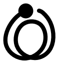
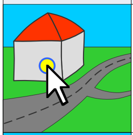
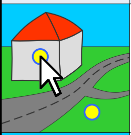
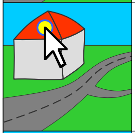
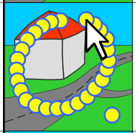
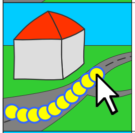
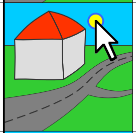
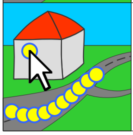
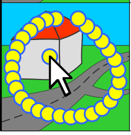
Sketch	Navigation Command	Sketch	Navigation Command	Sketch	Navigation Command
	Tilt up the camera.		Rotate the camera to right.		Take an overview position.
	Tilt down the camera.		Rotate the camera to left.		Undo last navigation.

Table 2. Examples of sketch-based navigation commands. Gestures can be used as modifiers for object-related sketches.

Sketch	Navigation Command & Result	Sketch	Navigation Command & Result
	**Point on a building:** Finding the shortest path to the building, driving there, and looking at the building.		**Point on the ground and point on a building:** Flying to the marked ground point and looking at the building finally.
	**Point on a building's roof:** Flying up to the roof, placing the camera on top and looking around.		**Point on the ground and circle-shaped gesture:** Flying to the marked ground point and looking around.
	**Curve on a street:** Driving along the street.		**Point on the sky:** Soaring above ground for overview.
	**Curve on a street and point on a building:** Driving along the street and looking at the building finally.		**Circle-shaped gesture and point on a building:** Soaring above ground and looking at the building finally.

In our approach, motion-oriented navigation commands are mainly defined by object-related sketches. Pen-based gestures cover user interface management operations (undo), low-level camera-oriented navigation commands (e.g., rotating or tilting), modify preceding object-related sketches, or trigger more complex navigations. Table 1 and Table 2 illustrate examples of sketchy navigation commands.

3.2 Navigation Affordances of 3D Objects

The navigation abilities and affordances of typical elements of 3D VEs play a key role in our approach. Taking into account the semantics of involved scene objects facilitates navigation strategies that can be adapted to specific users and tasks. These elements provide motion-oriented and task-oriented navigation affordances, as well as additional information that can be facilitated for the generation of appealing camera animations.

Thus, the sketch-based navigation technique requires for an appropriate model that not only contains geometry but also provides thematic information about the contained entities. Our implementation basis on the CityGML model, which is a specification for virtual 3D city and landscape models and supports, e.g., terrain models, vegetation models and detailed building models. The prototype implementation regards terrain, vegetation area, building, roof, street, and the sky as relevant object types.

4 Processing Sketch-Based Navigation Commands

The users enter object-related sketches by the left mouse button or by finger or stylus on touch-sensitive displays. For gesture input, the right mouse button or a view plane button are used. Additionally, sketch-based navigation processing comprises sketch recognition, sketch interpretation, and camera animation – see Fig. 2.

4.1 Interpreting Navigation Sketches

For the interpretation of navigation sketches, our technique takes into account the *sketch geometry* (curves and points), the *spatial context* (the virtual location to which the sketch is aligned or associated), and the *temporal context* (the sketch composition, command history, and drawing speed). [6]

Object-related sketches get evaluated from the scene graph by projecting each 2D sketch point (curves are discretized), calculating a set of ray intersections with the relevant objects, and retrieving the object types. For each sketch, its semantics is determined from the major type of the nearest to the camera intersection of each sketch point. All intersections corresponding to this type represent the projected sketch and form the basis for retrieving positions and orientations. Together with the geometry type, this semantics defines the meaning of the sketch, e.g. "Curve–Street".

Gestures are interpreted from their 2D shape without taking into account the underlying scenery. The sketched 2D geometry is analyzed by a shape recognition algorithm, which uses the distance and angle of intermediate gesture points as features for the correlation of drawn gestures and predefined template gestures and results in navigation commands such as "Gesture–CircleLeft".

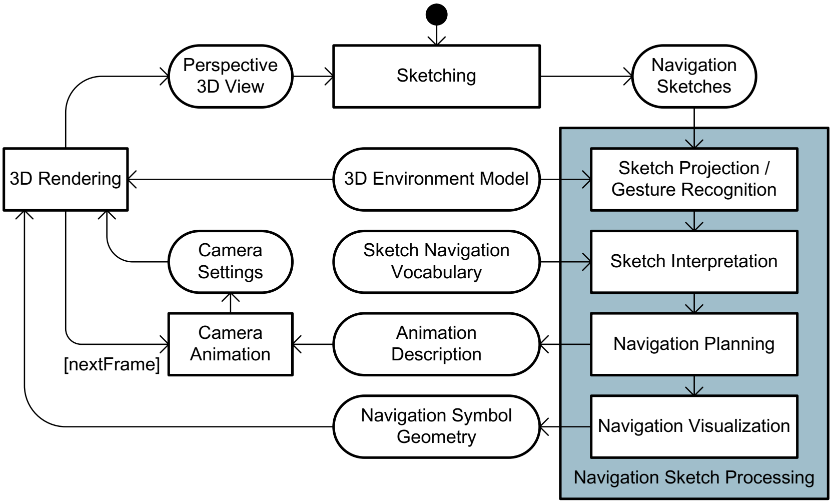


Fig. 2. Principal components of the sketch-based navigation command system

For composite sketches, the component interpretations are concatenated and thereby represent more complex navigation commands, e.g., “Curve–Street, Point–Building”. From these navigation command representations, the navigation system concludes how to generate camera animations from the sketch input.

To improve the usability of our sketch-based navigation, the history of navigation commands and animations is considered. For this, the navigation system stores the past navigation activities (the type of navigation and a possible target identifier). For example, a user points on a building and triggers the navigation “drive to building”. If the user points to the same building with the following sketchy navigation command, the system synthesizes a short camera flight around the object allowing the user to “inspect the building”.

4.2 Mapping Navigation Commands to Animations

Based on the determined navigation intention, the camera animation is planned. For each supported (composite) sketchy navigation command, the system features a handler comprising the knowledge of how to derive paths and orientations from projected sketches and gestures. For a curve on a street, the curve points are filtered for removing noise and interpreted as a path on that street. Extending this sketch by a point on a building, orients the camera toward the hit surface point. By contrast, a single “point on building” command leads to the computation of the shortest path to that building. The navigation handlers generate camera settings for key frames (e.g., starting point, intermediate points, and end point of a camera path), which are interpolated for creating the animation.

4.3 Visualization of Pending Navigation Commands

As a key element, our extended sketch-based navigation approach incorporates feedback to the users by visual cues about pending navigations. They act as a preview of how the system interprets the sketches, which navigation is determined, and allow the users to verify whether their navigation intention has been correctly recognized.

The navigation cues are integrated into the 3D scene and displayed and animated during the navigation animation. Path arrows on the terrain or street hint at where the camera will move along and billboard-attached target arrows indicate points of interest – see Fig. 3.

4.4 Sketching Speed

The speed at which sketch geometry is drawn can be utilized for improving the sketch-based navigation interface. It can denote the user experience, can be used for determining animation speed, and could influence the animation dramaturgy.

Sketching speed is different for near and far parts of the 3D scene. A path drawn at far distance has a smaller extend on the 2D view plane than a path

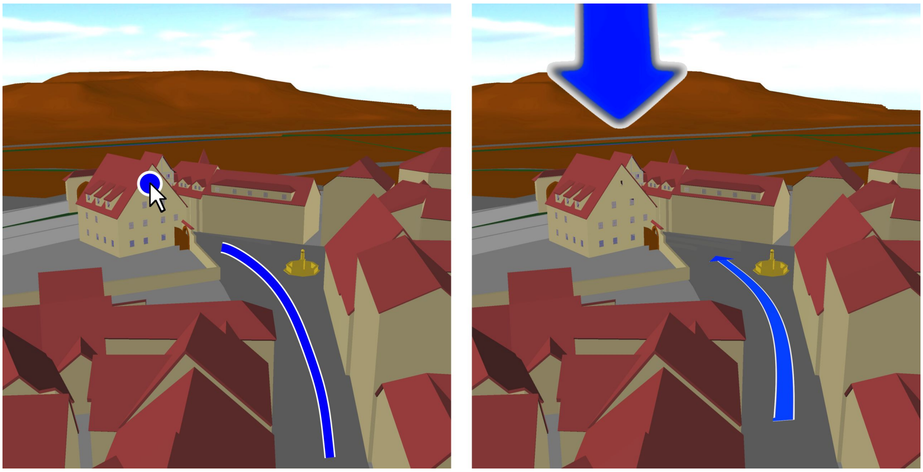


Fig. 3. Example of a sketch-based navigation command (left) and the resulting visual cues integrated in the 3D scene (right). The path arrow symbolizes the camera movement and the target arrow points to the selected building.

drawn nearby. Thus, the speed of object-related sketches is calculated from the path speed in 3D. As gestures are not projected into 3D, their speed is calculated from the speed of sketching on the view plane.

5 Conclusions and Future Work

The presented sketch-based navigation commands abstract the navigation process in 3D virtual environments. Instead of controlling and maneuvering the virtual camera, users rather specify their navigation goals, trigger automated navigation processes, and obtain smooth camera animations. The graphical navigation commands are interpreted according to shape, spatial context, and temporal context. The approach takes advantage of the inherent navigation affordances of the scene objects, considers sketching speed, and integrates visual feedback.

Sketch-based navigation lends itself for being used by non-experts on 3D navigation but also for providing task- and user-specific mechanisms to complex navigation operation. Furthermore, sketch-based navigation allows to implement a step-by-step interactive exploration of 3D VEs on thin clients (e.g., mobile devices), which would only handle the sketch input, while the corresponding server manages the 3D VE and renders the camera animations.

In future work, we will investigate task and goal-specific sketchy navigation commands and extend the sketch-based navigation vocabulary including more object types and navigation affordances for virtual 3D city and landscape models. Additionally, we plan to enhance the visual feedback mechanisms to return more information about recognized navigation intentions and to extend the camera dramaturgy taking into account additional 3D object types.

References

1. Buchholz, H., Bohnet, J., Döllner, J.: Smart and physically-based navigation in 3d geovirtual environments. In: IV 2005: Proceedings of the Ninth International Conference on Information Visualisation, Washington, DC, USA, pp. 629–635. IEEE Computer Society, Los Alamitos (2005)
2. Burtnyk, N., Khan, A., Fitzmaurice, G., Balakrishnan, R., Kurtenbach, G.: Stylecam: Interactive stylized 3d navigation using integrated spatial & temporal controls. In: UIST 2002: Proceedings of the 15th annual ACM Symposium on User Interface Software and Technology, pp. 101–110. ACM, New York (2002)
3. Cohen, J.M., Hughes, J.F., Zeleznik, R.C.: Harold: A world made of drawings. In: NPAR 2000: Proceedings of the 1st International Symposium on Non-Photorealistic Animation and Rendering, pp. 83–90. ACM, New York (2000)
4. Darken, R.P., Peterson, B.: Spatial Orientation, Wayfinding, and Representation. In: Handbook of Virtual Environment Technology, pp. 493–518. Lawrence Erlbaum Assoc., New Jersey (2002)
5. Döllner, J.: Constraints as means of controlling usage of geovirtual environments. Journal of Cartography and Geographic Information Science 32(2), 69–80 (2005)
6. Döllner, J., Hagedorn, B., Schmidt, S.: An approach towards semantics-based navigation in 3d city models on mobile devices. In: Gartner, M.P.P.G., Cartwright, W. (eds.) Location Based Services and TeleCartography. Lecture Notes in Geoinformation and Cartography, pp. 357–368. Springer, Heidelberg (2007)
7. Russo dos Santos, C., Gros, P., Abel, P., Loisel, D., Trichaud, N., Paris, J.P.: Metaphor-aware 3d navigation. In: INFOVIS 2000: Proceedings of the IEEE Symposium on Information Visualization 2000, Washington, DC, USA, 2000, pp. 155–165. IEEE Computer Society, Los Alamitos (2000)
8. Hachet, M., Decle, F., Knödel, S., Guitton, P.: Navidget for easy 3d camera positioning from 2d inputs. In: IEEE Symposium on 3D User Interfaces 2008, March 2008, pp. 83–89 (2008)
9. Hanson, A.J., Wernert, E.A.: Constrained 3d navigation with 2d controllers. In: VIS 1997: Proceedings of the 8th Conference on Visualization 1997, pp. 175–182. IEEE Computer Society Press, Los Alamitos (1997)
10. Igarashi, T., Kadobayashi, R., Mase, K., Tanaka, H.: Path drawing for 3d walkthrough. In: UIST 1998: Proceedings of the 11th annual ACM Symposium on User Interface Software and Technology, pp. 173–174. ACM, New York (1998)
11. Igarashi, T., Matsuoka, S., Tanaka, H.: Teddy: A sketching interface for 3d freeform design. In: SIGGRAPH 1999: Proceedings of the 26th annual Conference on Computer Graphics and Interactive Techniques, New York, NY, USA, pp. 409–416. ACM Press/Addison-Wesley Publishing Co. (1999)
12. Khan, A., Komalo, B., Stam, J., Fitzmaurice, G., Kurtenbach, G.: Hovercam: interactive 3d navigation for proximal object inspection. In: I3D 2005: Proceedings of the 2005 Symposium on Interactive 3D Graphics and Games, pp. 73–80. ACM, New York (2005)
13. Tan, D.S., Robertson, G.G., Czerwinski, M.: Exploring 3d navigation: Combining speed-coupled flying with orbiting. In: CHI 2001: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, pp. 418–425. ACM, New York (2001)
14. Zeleznik, R.C., Herndon, K.P., Hughes, J.F.: Sketch: An interface for sketching 3d scenes. In: Rushmeier, H. (ed.) SIGGRAPH 1996 Conference Proceedings, pp. 163–170. Addison Wesley, Reading (1996)

Adaptive Layout for Interactive Documents

Kamran Ali[1], Knut Hartmann[2], Georg Fuchs[1], and Heidrun Schumann[1]

[1] Institute of Computer Graphics
Department of Computer Science, University of Rostock
Albert Einstein Str. 21, D-18059 Rostock, Germany
{kamran.ali,georg.fuchs,schumann}@informatik.uni-rostock.de
[2] Flensburg University of Applied Science, Germany
knut.hartmann@fh-flensburg.de

Abstract. In many application domains there is a strong need to produce content both for traditional print media and for interactive media. In order to fully benefit from digital devices, online documents must provide mechanisms to support interactivity and for the personalization of content. Thus, powerful authoring tools as well as flexible layout techniques are needed to display dynamic information effectively. In this paper we present a novel approach to template-based design of complex non-grid layouts and propose a unique combination of constraint-based and force-based layout mechanisms. Our authoring tool enables designer to specify designs pattern with overlapping content elements that are transformed into layout constraints and force systems by the layout engine. The smooth integration of interactive illustrations into complex layouts also induces constraints to achieve temporal coherent layout. We propose algorithms to achieve graceful layouts even when the space consumption of content elements significantly differs from the defaults specified in layout templates.

1 Introduction

Rapid growth of internet and advanced web technologies has seen large volumes of information being delivered over web. In digital networked world content can be frequently updated and individuals may request information in different forms at different devices and have their own viewing preferences. Therefore dynamic requirements of the digital media restrict that the documents cannot be formatted in advance. Hence, the traditional processes of content creation, editing, and quality assurance have to be completely revised in digital environments.

The reuse of printed material (e. g., text books, encyclopedia, or technical documentation) in interactive applications is very promising due to the high quality of their content. But layout techniques in current web browsers are limited in scope and quality. Besides, more often that not information recipients are going to be many individual users who are anonymous and connected to different devices. Variations in display resolution and computing power of electronic devices may require that the document is presented differently for each reader. Under this scenario, static design of a document might become unsatisfactory when the content or device capabilities change.

A. Butz et al. (Eds.): SG 2008, LNCS 5166, pp. 247–254, 2008.

The situation becomes even harder for layouts with many illustrations. While web browsers guarantee the readability of text on all devices and offer at least some room to interact with text, illustrations are inflexible elements in the layout. *Interactive illustrations* on the other side would offer new possibilities for online documents. 2D and 3D visualizations can be overlayed with secondary elements (e. g., textual annotations, anchor points, connecting lines, scales, legends etc.). In addition, an unequal scaling of imagery and secondary elements can guarantee the readability of textual annotations and updation of content within textual annotations.

In this paper, we propose tools that (i) enable authors / designers to create blueprints for complex layouts that integrate interactive illustrations and (ii) describe several layout techniques to render documents with dynamic content. This paper is organized as follows: Sec. 2 reviews the related work. We present insights into our novel approach to adaptive document layout in Sec. 3. Sec. 4 gives a case study. Finally, Sec. 5 summarizes the initial results of our work and discusses directions of future work.

2 Related Work

Earlier research on document layout mainly focused on line-breaking and pagination. Commercial desktop publishing systems also provide a set of pre-designed layout templates. But these systems aim at static documents and do not provide any mechanism to author or layout documents with dynamic content. In contrast, the render engines of current web browsers were designed to re-layout the document's content dynamically. But there are still few mechanisms for authors to adjust the design of the layout to the capabilities of different devices.

Current research in the field of automated document layout has mainly explored graphical constraints and typographic grids.

Constraints: Weitzmann's system [16] and Graf's LayLab [10] use grammars to specify constraints of two types: abstract constraints (e. g., description-of, author-of etc.) and spatial constraints (e. g., right-of, top-aligned etc.). Spatial constraints can be directly used as input to the constraint solver. Abstract constraints must be converted to spatial constraints before the layout resolution might take place.

As powerful and expressive as these grammars are, it becomes extremely difficult to describe a set of all possible high-level relationships between the items of a presentation. But the major problem of this technique is the treatment of contradicting constraints that prevents any solution.

Grids: Display area of the presentation is split into an array of upright rectangular cells separated by margin space [14]. Each layout element has to span an integral of cells in width and height. Hence, the underlying typographic grid works as a proportional regulator for content items. Feiner's approach [6] imposes spatial constraints on the layout items via a typographic grid to generate presentation. The underlying grid is generated automatically from document's content and viewing restrictions.

By encoding the desired properties of a layout in a set of layout templates, Jacobs et. al. [11] were able to adjust the layout of online documents to the capabilities of different devices. Each layout template specifies abstract and spatial constraints on the components of the presentation. Several techniques are applied to select and adapt layout templates from the set of available templates. Their system also performs pagination to split the document's content on multiple pages that in turn are formatted separately. Even though this approach achieves impressive results for static content, it was not designed for interactive media elements. Moreover, the strict-grid based design may not be suitable for documents containing large number of *cut-out* illustrations.

3 Adaptive Document Layout

3.1 Architecture Overview

Figure 1 shows the architecture of our adaptive document system. We provide a template authoring tool that the designers can use to design document templates for various viewing scenarios (e. g., different screen aspect ratios or device classes). Designers can define layout elements by drawing rectangular boxes or, for complex non-grid layouts, arbitrary shaped convex polygons on the screen. The element placement is controlled using linear arithmetic equalities and inequalities constraints. For each template, designers also dictate when the design is appropriate by specifying which content must be present and under which viewing conditions the template can be used. The templates are stored in a *template library*.

Content authors prepare the document contents by tagging individual content items. These include text, static illustrations, animations, and interactive 3D representations. Items can be tagged with their importance values. The authors may also define item priorities which are considered to resolve space requirements on screen.

During the layout process, the *layout engine* determines the appropriate set of templates in the template library by matching content tags to the template item tags. For the selected templates, it adjusts the size and position for each layout element using

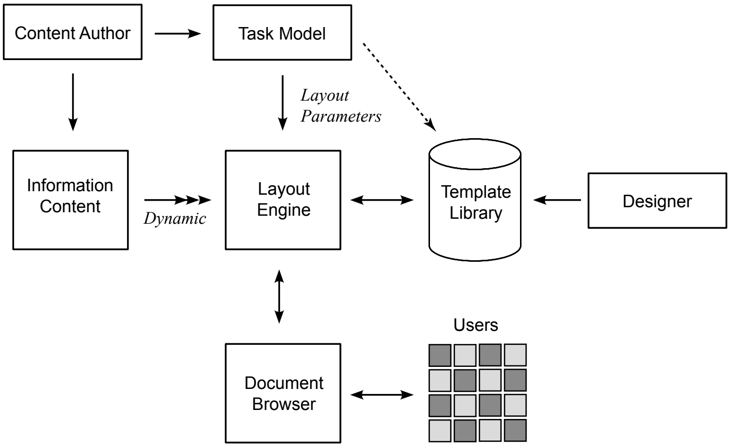


Fig. 1. Adaptive document system design

constraint solving methods. The constraint-based layout is then further refined in a post-processing phase. We discuss this most important part of our approach in Section 3.2.

3.2 Layout Engine

Document layout phase is divided into two stages. In the first stage an initial layout is derived from templates using the given constraints. The second stage is post-processing stage; forced-directed methods are applied on selective regions to modify the initial layout. During the latter phase, a check-mechanism is enabled so that layout adjustments are made only when they don't conflict with template constraints.

Initial Layout: Constraint-based document layout technique [3] are used to compute the initial layout configuration. Layout engine loads templates from the template library and evaluates them. It matches the template items to the given content and computes how well this content fits within the template. For the selected template, size and position variables of each element in the template are set by the layout engine. These values are fed into the constraint solver for resolving a given set of constraints. In our system we use CASSOWARY TOOLKIT [2] which is an incremental linear arithmetic constraint solver. Since the constraint-based layouts are a well explored area, in this paper we focus our attention on the forced-directed layout methods.

Post-processing: The post-processing phase is used to modify the initial configuration set by constraint-based layout. It is especially helpful to create complex non-grid layouts for varying amount of illustrative material which cannot be laid out using the given template. We explore force-directed layout techniques to provide graceful layout mechanism even when the content type and size is significantly different from what is suitable for the given template.

We formulate document layout as a *graph-layout* problem. A graph $G = (V,E)$ consists of vertices V and a set of edges E where each edge connects a pair of vertices. Fruchterman and Reingold [7] proposed spring-embedding approach for drawing graphs [4]. According to this, an attractive force enables the vertices to attract each other if they are edge-connected, thus keeping them close together. A repulsive force causes the vertices to repel each other, thus avoiding their overlapping.

Let a graph G represent a document. Vertices denote content items, typographic guidelines, and area boundary lines. Edges are used to model relationships between vertices. For example an edge can model dependency between two content items so that they are placed closer to each other. An edge between a typographic grid line and a content item can be used to indicate the alignment of item. Similarly an item can be attached to its reference position using an edge. In contrast, a repulsive force would ensure that the items mind a certain distance between them.

Types of forces: In our force-directed approach, several forces are applied on objects:

- An attractive force between related elements; *(derived from item dependency list)*
- Attractive force between the current and reference placement of an object, *(derived using the template)*
- Repulsive force between elements; *(derived from margin and gutter)*

- Repulsive force at the page/screen boundary; *(derived from page margin)*
- Attractive force between previous and current positions of an object in successive layouts.

If f_a and f_r are the attractive and repulsive forces, respectively, with d distance between the objects, then $f_a(d) = d^2/k$ and $f_r(d) = -k^2/d$. Here k is the ideal distance between two objects where attractive and repulsive forces cancel each other. The magnitude of these forces is dependent on the distance between the objects. Minimum distance between the two objects is determined by anti-podal pair between the convex-shaped polygons. The layout engine determines attractive and repulsive forces on each object i in the layout. These forces are summed together to calculate the magnitude and direction of displacement for i. At run-time content elements are attracted toward their old positions in the previous layout configuration. This allows us to achieve temporal coherence over the successive layouts.

A pressure-directed part controls different object sizes to objects. According to Boyle's law, volume of the gas increases when the pressure decreases at constant temperature and vice versa, i. e., the volume decreases when the pressure is increased. This relationship between pressure P and volume V is given as $P \times V = constant$.

To control the resizing of elements in layout, we incorporate *Inner* and *outer* pressure forces in our system with a modification to the approach mentioned in [12].

- The more relevant a content object, the higher the inner pressure, and vice versa;
- The more free space on the screen, the lower the outer pressure, and vice versa;

More pressure inside an element than the outer pressure leads to an enlargement of the element and decrease in inner pressure and to rise of outer pressure. Conversely more outer pressure than the inner pressure leads to the reduction of object. With this mechanism free space left by missing content items can be utilized by the other items. Items can be added/removed and resized freely on the screen as the pressure-directed method manages space requirements. This paves the way to integrate adaptive illustrations into complex layouts which impose and obey global layout requirements.

Our force-directed layout approach is given in Algorithm 1. It extends force-based algorithms in [7] used for graph drawing and [12] intended for placement of rectangular windows. In contrast, we apply force-directed algorithms for document layout. Our approach analyzes template constraints to infer the system of forces. Elements can be arbitrary shaped convex polygons overlapping each other. Moreover large amounts of content can be split into several pages that are laid out individually.

Using template authoring tool designers can define the size of items and specify if resizing is allowed by the pressure-directed forces. They can also declare a set of constraints that the post-processing must obey in the final layout. Initial layout sets the system of pressure forces in balance which are then manipulated at run-time. Since the pressure-directed resizing scheme goes hand-in-hand with force-directed displacement scheme, layout conflicts are resolved dynamically. Layout modifications in the post-processing step are applied only when they don't lead to an invalid layout regarding the constraints.

Algorithm 1. Force Directed Layout

```
{determine initial placement using templates}
for $i = 1$ $to\ iterations$ do
  {force-directed part}
  for $v \in V$ do
    {calculate repulsive force $f_r$ and attractive forces $f_a$ upon each element $v$}
    {move element $v$ in the direction of $f_r + f_a$}
    {update forces}
  end for
  {pressure-directed part}
  for $v \in V$ do
    if $v.pressure_{inner} > v.pressure_{outer}$ then
      enlarge element $v$
    else
      reduce element $v$ in size
    end if
    update pressure values
  end for
end for
```

Once the page layout is computed, content can be flowed into each determined region in the layout. The formatted document is presented via document browser to the users who can interact with it. For larger amount of content, we need a *paginator* to split the content into a set of discrete pages, subject to various constraint. Optimal pagination is a complex issue and begs more attention. Due to limited space in this paper we refer the readers to study pagination task in [15].

Our layout engine also incorporates an integrated illustration system to support dynamic illustrations in documents. Efficient dynamic labeling algorithms [1] and [8] compute placement of textual annotations on images. The labeled illustrations are automatically adapted to fit within 2D regions allocated to them by the layout engine.

3.3 Task-Driven Document Layout

Up to now, we only considered aspects regarding the layout of individual dynamic documents. However, their suitability as a means to effectively and concisely communicate complex structure make dynamic documents a primary choice for online documentation such as technical manuals for mobile maintenance support ("e-Manuals") [8,9]. Tasks associated with repair and maintenance work typically follow laid down procedures, which lend themselves to explicit notation in the form of a *task model* [13]. Generally, a task model forms a hierarchy of compound tasks with subtasks and conditional/temporal relationships between them.

This task model can be used to store additional information, to guide and optimize the initial layout of dynamic documents with respect to the current task at hand (cf. Fig. 1). To this end, the content author specifies, for each task, which template from the library is preferable for the content associated with that task. Moreover, she can specify different priorities per subtask for content items that will affect how the content is laid out (cf. Sect. 3.2). This enables the author to specify which aspect(s) of the document are most important for the current working step. Note that after the task-driven generation of *initial* document views, interaction by the user still allows for dynamic layout changes as described in Section 3.2.

4 Case Study

Figure 2 shows the results of our forced-directed layout approach overlayed with the forces used to compute the layout. The content has been laid out in the style of a VISUAL DICTIONARY book [5]. In order to match the content of the descriptive text and illustrations, all figures are richly annotated with text labels.

Note also, that the bounding rectangles for the primary visual elements and their associated secondary elements overlap. The layout was achieved by (i) placing all primary elements at reference positions in template and deciding for appropriate scaling factor for the individual images or image groups and (ii) placing all the remaining secondary elements. Thus, secondary elements of all depictions share the unused background as a common resource. This layout procedure is based on semantic considerations (relevance of layout elements), from structural constraints (common properties for all elements of a group) and the difficulty to reposition or to adjust an layout element. Textual annotations, for example, *can* easily float and the line-break of the descriptive text can be adjusted in order to float around the thumbnails images.

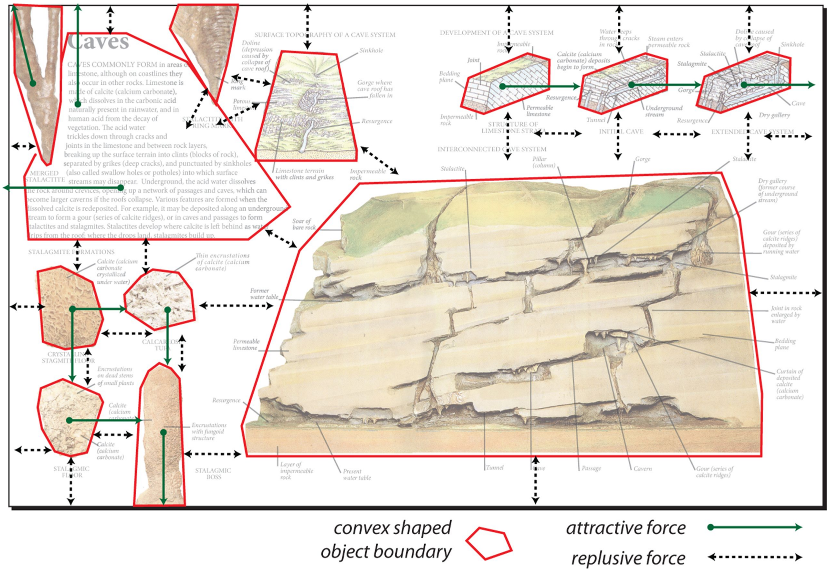


Fig. 2. Forced based layout

5 Conclusion and Future Work

This paper proposes novel strategies to adapt the layout of multimodal documents under different viewing conditions according to the changes in content and user interaction. There is still room for many improvements. First, our algorithm should exploit more parameters of the layout: font sizes, line width, column size, or the grid size. Moreover, we believe that this freedom to adjust all parameters that determine the overall layout is the only way to convert aesthetics and readability criteria into metrics, forces, and constraints. Another interesting aspect is the automatic determination of layout parameters or layout templates from scanned documents. This would ease the time-consuming and error-prone trial&error process of fiddling around with layout parameter to providing a set of good examples. Of course, proper evaluations are needed both for the authoring tool as well as for the dynamic layout algorithms.

References

1. Ali, K., Hartmann, K., Strothotte, T.: Label Layout for Interactive 3D Illustrations. Journal Of The WSCG 13, 1–8 (2005)
2. Badros, G.J., Borning, A., Stuckey, P.J.: The Cassowary Linear Arithmetic Constraint Solving Algorithm. ACM Trans. Comput.-Hum. Interact. 8(4), 267–306 (2001)
3. Borning, A., Lin, R., Marriott, K.: Constraint-Based Document Layout for the Web. Multimedia Systems 8(3), 177–189 (2000)
4. Di Battista, G., Eades, P., Tamassia, R., Tollis, I.G.: Graph Drawing: Algorithms for the Visualization of Graphs. Prentice Hall, Englewood Cliffs (1999)
5. Docherty, P. (ed.): DK Ultimate Visual Dictionary. Dorling Kindersley Publishing (2002)
6. Feiner, S.K.: A Grid-Based Approach to Automating Display Layout. In: Graphics Interface 1988, pp. 192–197 (1988)
7. Fruchterman, T.M.J., Reingold, E.M.: Graph Drawing by Force-Directed Placement. Software — Practice and Experience 21(11), 1129–1164 (1991)
8. Fuchs, G., Luboschik, M., Hartmann, K., Ali, K., Schumann, H., Strothotte, T.: Adaptive Labeling for Interactive Mobile Information Systems. In: 10th Int. Conf. on Information Visualisation (2006)
9. Fuchs, G., Reichart, D., Schumann, H., Forbrig, P.: Maintenance Support — Case Study for a Multimodal Mobile User Interface. In: Multimedia on Mobile Devices II, SPIE, vol. 6074 (2006)
10. Graf, W.H.: Constraint-Based Graphical Layout of Multimodal Presentations. In: Int. WS on Advaned Visual Interfaces, pp. 365–385 (1992)
11. Jacobs, C., Li, W., Schrier, E., Bargeron, D., Salesin, D.: Adaptive Grid-Based Document Layout. ACM Transactions on Graphics 22(3), 838–847 (2003)
12. Lüders, P., Ernst, R., Stille, S.: An Approach to Automatic Display Layout Using Combinatorial Optimization Algorithms. Soft. — Prac. & Exp. 25(11), 1183–1202 (1995)
13. Mori, G., Paterno, F., Santoro, C.: CTTE: Support for Developing and Analyzing Task Models for Interactive System Design. IEEE Trans. Soft. Eng. 28(8), 797–813 (2002)
14. Müler-Brockman, J.: Grid Systems in Graphic Design. Verlag Arthur Niggli (1996)
15. Plass, M.F.: Optimal Pagination Techniques for Automatic Typesetting Systems. PhD thesis, Department of Computer Science, Stanford University (1981)
16. Weitzmann, L., Wittenburg, K.: Grammar-Based Articulation for Multimedia Document Design. Multimedia Systems 4(3), 99–111 (1996)

Curvature- and Model-Based Surface Hatching of Anatomical Structures Derived from Clinical Volume Datasets

Rocco Gasteiger, Christian Tietjen, Alexandra Baer, and Bernhard Preim

Otto-von-Guericke University of Magdeburg,
Department of Simulation and Graphics, Germany
gasteiger@isg.cs.uni-magdeburg.de

Abstract. We present a texture-based method to hatch anatomical structures derived from clinical volume datasets. We consider intervention planning, where object and shape recognition are supported by adding hatching lines to the anatomical model. The major contribution of this paper is to enhance the curvature-based hatching of anatomical surfaces by incorporating model-based preferential directions of the underlying anatomical structures. For this purpose, we apply our method on vessels and elongated muscles. Additionally, the whole hatching process is performed without time-consuming user interaction for defining stroke direction and surface parameterization. Moreover, our approach fulfills requirements for interactive explorations like frame coherence and real time capability.

1 Introduction

For intervention planning, e.g., surgery or interventional radiology, medical doctors need a precise mental understanding of the anatomy of the particular patient. The planning is performed on individual patient datasets with reduced resolution and noise artifacts, which impedes the analysis more complicate. For this purpose, hatching offers a great potential because it is traditionally used for shading in anatomical illustrations, facilitating structure distinctions, and describing local shape characteristics. The latter one was investigated by Kim et al. [1] who have shown that lines along curvature directions improve the human shape perception compared to a simple shading or arbitrary directions. Applied to semi-transparent surfaces, these lines expose internal objects where important shape characteristics like concave and convex regions are still noticeable. Due to the characteristics of anatomical models and the quality of datasets, common hatching approaches which are only based on curvature information, are not appropriate.

In this paper we focus on hatching of anatomical surfaces within intervention planning. We consider vessels and elongated muscles and derive model-based preferential directions. These directions are combined with curvature information to determine the final hatching direction. Since intervention planning is a routine

A. Butz et al. (Eds.): SG 2008, LNCS 5166, pp. 255–262, 2008.

task, we aim at a non-manual specification of hatching directions and no time-consuming parameter settings. Furthermore, we require frame coherence and real time capability for interactive exploration. Based on the resulting direction field, we perform a texture-based approach to apply line textures to the anatomical structures. This leads to a new contribution of shape enhancement within the intervention planning.

2 Related Work

Existing hatching techniques can be classified as image-, object- and texture-based approaches. Hertzmann and Zorin [2] compute principle curvature directions and introduce an optimization method to create geodesic streamlines by projecting them into the image space. Additionally, the authors point out that many parameter settings are necessary for their rendering result. Rössl and Kobbelt [3] additionally apply user-defined reference directions in the image plane to align the curvature directions. Ritter el al. [4] generate hatching strokes for smoothed vessel surfaces by calculating differences of several depth buffer images. Although they produce convincing results, frame coherence is not ensured by these methods. For object-based methods hatching lines are completely generated onto the 3D surface of an object. Most of the object-based approaches also compute principle curvature information for each vertex and align polygonal lines to them [5], [6], [7]. Zander et al. [7] use an extension of the optimization method from [2] to smooth the vector field. Since the lines are attached to the surface, frame-coherent renderings are achieved. However, hatching lines are only aligned to curvature directions. In addition, [6] have limited real time capabilities and [7] involves complex parameter settings to depict contrast images. Deussen et al. [8] introduce non-curvature alignment by generating lines by intersecting the geometry with a set of individually placed planes. The computation of a geometric skeleton allows to automatically determine the orientation of the intersection planes. However, the skeleton algorithm is not particularly appropriate for complex or branching objects like vessel structures. The main challenge for texture-based approaches is to determine appropriate texture coordinates for mapping 2D images onto a complex 3D surface with low distortions. Praun et al. [9] introduce *lapped patches* to combine local parameterization with local alignment to an underlying vector field. They use texture blending to reduce texture seams at patch boundaries and to perform real time capability and frame coherence. They extend their approach with *Tonal Art Maps* (TAM) for a light and view depending hatching with spatial and temporal coherence [10].

3 Requirement Analysis

Hatching of common anatomical structures for intervention planning is a challenging task. Several approaches may generate pleasant results, but lack one or more of requirements for use in intervention planning. Most methods align hatching strokes only along curvature information or require some user interaction to place reference lines. However, curvature information do not describe model-based directions, such fiber directions of a muscle. Furthermore, model-based directions cannot be extracted directly from images of common clinical CT or MRI data. Instead, a surface description has to be derived. For anatomical structures common representation are polygonal representations. In addition to curvature information, model-based directions have to be extracted automatically from these surfaces. The surfaces are extracted from binary segmentation masks, which need to be smoothed to eliminate the staircase artifacts. The remaining artifacts lead to a complex surface with respect to the number of convex and concave vertices. Therefore, correct curvature estimations suffer from these artifacts.

We employ a texture-based technique since it is as fast as image-based techniques and frame-coherent like object-based approaches. The quality of the texture mapping strongly depends on the underlying surface parameterization. Since we are interested in the local alignment of hatching strokes, we choose a modification of *lapped patches* [9] as an appropriate surface parameterization.

4 Curvature- and Model-Based Vector Field

In the following we describe an approach to automatically generate a curvature- and model-based vector field for vessel structures and elongated muscles. Based on our observations from anatomic textbooks, vessels and muscles have a global specific preferential direction. The hatching lines run radially around vessel surfaces and along the fiber direction for muscle structures. For organs, there exists no such model-based direction. The illustrator chooses the line orientation that depicts the surface shape most suitable from the current viewpoint. Hence, we do not consider this type of anatomical structure in the paper. It is noticeable that curvature information are not sufficient to generate all aforementioned model-based directions. Although the radial alignment for vessel structures theoretically corresponds to the first principle curvature direction, this does not hold for rough vessel surfaces derived from routine medical image data. The corresponding directions contain noise that has no radial orientation. Furthermore, fiber directions cannot be described by curvature, which also introduces singularities (see Fig. 1(a)).

In the following, we perform three steps to automatically get a shape enhanced, frame-coherent and almost non-singular model-based vector field depending on the underlying anatomical surface:

1. Adaptive curvature estimation (Fig. 1(a)),
2. Definition of a model-based preferential direction (Fig. 1(b)), and
3. Combination of 1 and 2 to the final vector field (Fig. 1(c)).

The latter one resolves singularities for vessels and muscles because of their inherent preferential direction. For automatic determination for which anatomical structure its preferential direction has to compute, the structure type (vessel,

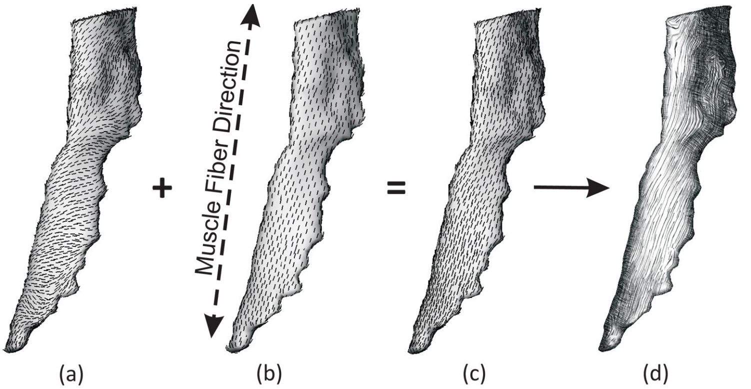


Fig. 1. Pipeline of our hatching approach: (a) curvature estimation, (b) approximation of preferential direction (depicted as arrow), and (c) combination of (a) and (b) to the final model-based vector field, (d) employing TAM textures aligned to the vector field

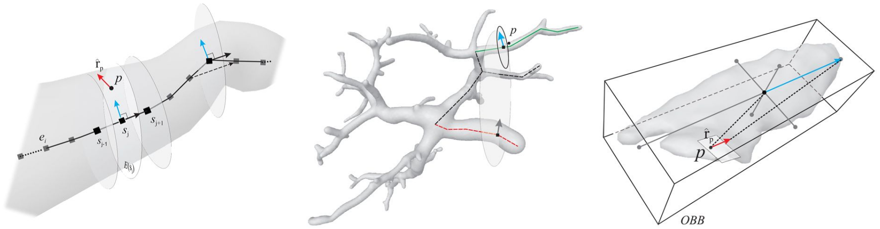
muscle) is stored to the dataset during the prior segmentation process. Finally, we apply hatch textures (e. g., TAM's) onto the surface with lapped patches (Fig. 1(d)). For a more precise description of each processing step, we refer to our technical report [11]. Additionally, more results are presented therein.

4.1 Adaptive Curvature Approximation

At first we apply a Laplace filter to smooth a copy of the anatomical surface. On this instance, we estimate the curvature direction used for each original vertex p. Due to the simple surface topology (number of convex and concave vertices) of muscle structures, we select a quadratic surface fitting method with topology-independent parameterization according to [12]. Vessel structures exhibit a more complex surface where a topology-independent parameterization does not always preserve the ordering of neighbors around p. Hence, we select a topology-dependent parameterization by approximation of geodesic polar coordinates of each p according to Rössl et al. [5].

4.2 Model-Based Preferential Directions

In order to obtain model-based preferential directions, we employ the internal skeleton of each structure to derive their preferential directions (inspired by Deussen et al. [8] and Roessl et al. [3]). For **vessel structures**, we extract the internal skeleton by the skeletonization algorithm presented in [13]. Based on the segmented voxel dataset this method combines topological thinning with distance transformation. With an appropriate sensitivity parameter, irrelevant side branches are effectively removed. The result of the skeletonization is a graph G which contains the edges e_i as connections between different branchings. Each edge consists of individual skeleton voxels s_j.



(a) (b) (c)

Fig. 2. Definition of the preferential direction $\hat{\boldsymbol{r}}_{\boldsymbol{p}}$ at a vertex p by internal skeletons for (a, b) vessel and (c) muscle structures: (a) Projection of the ”‘up”’vector (blue) of the corresponding intersection plane $E(s_j)$ (b) into the tangent plane of p as $\hat{\boldsymbol{r}}_{\boldsymbol{p}}$ (red), (c) Approximation of the skeleton by the OBB and choosing the longest axis (blue) as $\hat{\boldsymbol{r}}_{\boldsymbol{p}}$

The local radial alignment for vessel structures corresponds to the direction which is perpendicular to the local direction of the skeleton course. To compute such a vector, we need the nearest local skeleton direction for each vertex. Fig. 2(a) illustrates a vessel section with its corresponding edge e_i and some skeleton voxels as black squares. Through each s_j we place a plane $E(s_j)$ (circular disk) defined by s_j and its direction vector (black arrow). The latter one is calculated by central differences of its two incident skeleton voxels $\|s_{j+1} - s_{j-1}\|$. Additionally, we determine the nearest plane $E(s_j)$ to p. The ”up”-vector (blue arrow) that is perpendicular to the direction vector, lies within $E(s_j)$ and describes the local radial direction of p. The projected vector of this direction to the tangent plane of p is the local (normalized) preferential vector $\hat{\boldsymbol{r}}_{\boldsymbol{p}}$ for p. For complex vessels we firstly determine the corresponding skeleton edge e_i for each vertex p before the nearest plane is selected. This requirement is illustrated in Fig. 2(b) where the vascular tree of the liver is shown. There are two planes nearby at p, but only the green line with the bordered plane, is the corresponding edge to p to get the right local ”up”-vector.

For **elongated muscle structures**, we have to define the fiber direction which indicates the course between origin and onset of the muscle. We simply approximate this orientation by the longest principle axis of the object-oriented bounding box (OBB). Fig. 2(c) shows a muscle structure surrounded by its OBB as well as the corresponding principle axes. Among all three directions the vector $\boldsymbol{u}$ is the longest one and we define $\hat{\boldsymbol{r}}_{\boldsymbol{p}}$ as the projection of $\boldsymbol{u}$ to the tangent plane of each p.

4.3 Combination of Curvature and Preferential Direction

With the aforementioned approach, we obtain two kinds of direction information for vessel and muscle structures at each vertex p: curvature information in terms of both principle directions, and preferential direction $\hat{\boldsymbol{r}}_{\boldsymbol{p}}$ depending on the underlying anatomical structure. In the following, we combine this information to a stable vector field, which takes local curvature behavior into account. In Fig. 3(a)

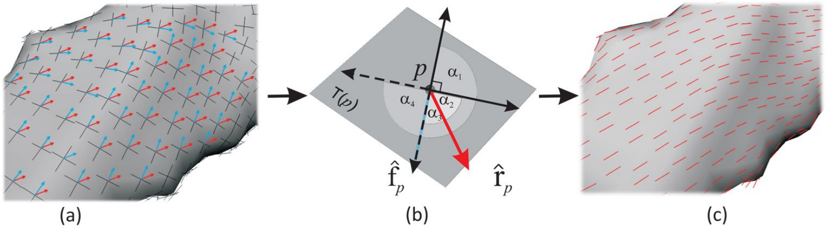


Fig. 3. (a) Curvature information (crosses) and preferential direction (red arrow=$\hat{\boldsymbol{r}}_{\boldsymbol{p}}$) at each vertex p, (b) Selection of the principle direction which has the smallest angle to $\hat{\boldsymbol{r}}_{\boldsymbol{p}}$ and denote it as $\hat{\boldsymbol{f}}_{\boldsymbol{p}}$, (c) Resulting and optimized model-based vector field

for each vertex its two principle directions are shown as black crosses and one preferential direction is shown as red arrow. Fig. 3(b) illustrates all directions for one vertex p within its tangent plane $T(p)$. Additionally, the enclosed angles between $\hat{\boldsymbol{r}}_{\boldsymbol{p}}$ and all four possible principle directions are included. For the initial vector field, we select the principle direction which has the smallest angle α_i ($i \in 1, 2, 3, 4$) to $\hat{\boldsymbol{r}}_{\boldsymbol{p}}$ and denote this direction as $\hat{\boldsymbol{f}}_{\boldsymbol{p}}$, indicated by a blue arrow.

The resulting vector field has still discontinuities (Fig. 3(a)), where each selected principle direction is highlighted (blue). We improve the alignment of each $\hat{\boldsymbol{f}}_{\boldsymbol{p}}$ to the preferential direction $\hat{\boldsymbol{r}}_{\boldsymbol{p}}$ and to its neighborhood. This means we aspire to a global geodesic course of the vector field. We use an energy minimization problem. For each optimization step, we rotate $\hat{\boldsymbol{f}}_{\boldsymbol{p}}$ into the direction of $\hat{\boldsymbol{r}}_{\boldsymbol{p}}$ and calculate the average angular deviation to all $\hat{\boldsymbol{f}}_{\boldsymbol{i}}$. The rotation stops, if the angle between $\hat{\boldsymbol{f}}_{\boldsymbol{p}}$ and $\hat{\boldsymbol{r}}_{\boldsymbol{p}}$ as well as the angular deviation becomes larger in comparison to the previous step. For each vertex we perform this operation recursively until the optimization is minimized. Finally, we compute a direction vector for every face by averaging the directions of the face vertices. Since we prefer a line adjustment that is more aligned to the preferential direction, the aforementioned simplified optimization method is sufficient. Fig. 1(c) and Fig. 3(c) show results after the optimization. A geodesic vector field is computed without singularities, aligned to the preferential direction, and includes local curvature behavior.

5 Results and Discussions

All presented results do not need any user interaction and were processed on a Pentium 4 processor with 3.2GHz and NVIDIA Quadro FX2000 GPU. Depending on mesh resolution each dataset needs some preprocessing time to calculate the vector field and surface parameterization, which is noticed for each example. During the rendering, we obtain real time capability and frame coherence. Computed texture coordinates are stored to each surface for further renderings.

Fig. 4(a) shows the resulting hatching on a neck vessel (preprocessing of 10 seconds with 878 faces) with a simple line texture. As shown in the rendering, the hatching lines have a radial orientation, and in spite of the rough surface no

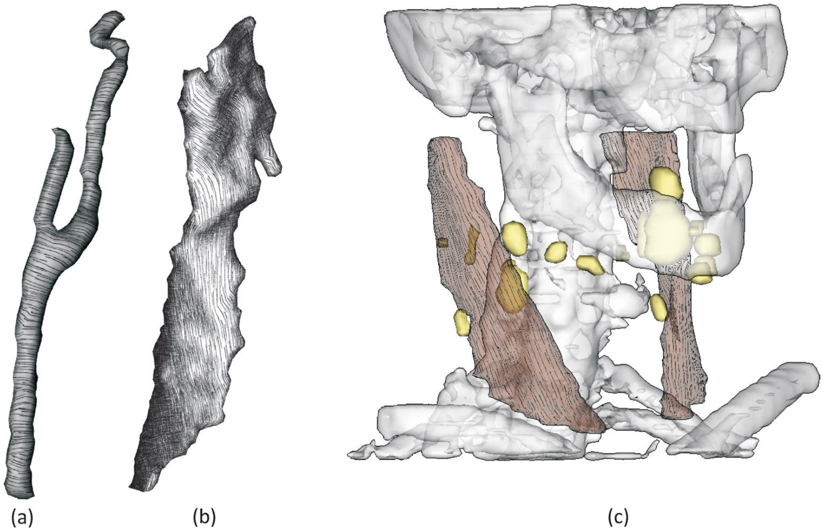


Fig. 4. (a) Hatching a neck vessel with a simple line texture, (b) Our approach applied with TAM textures to a neck muscle, (c) Opacity mapping of simple line textures to two neck muscles within a scene of neck surgery planning

strong texture distortions are visible. The TAM concept is employed in Fig. 4(c) for a neck muscle (preprocessing of 30 seconds with 2749 faces) to depict light conditions. It is noticeable that TAM texturing supports the depiction of convex and concave regions. However, for regions with low illumination the dark tones hide surface characteristics, noticeable at the shape boundaries of the muscle. Fig. 4(c) shows a scene of neck surgery planning. Surgical decisions for neck dissections are depending on the relative position between lymph nodes (yellow) and muscles (brown). Hence, lymph nodes as focus structures are opaque, bones (gray) and muscles are semi-transparent. Additionally, all structures are highlighted with their silhouettes. We applied a simple line texture with opacity mapping on the muscles, generated with our hatching approach. It needs a one-time preprocessing of 30 seconds for both muscles (5620 faces). Lymph nodes behind the muscles are still visible, but muscle surfaces obtain a more precise description. Additionally, the fiber direction is depicting and facilitates the structure distinction to surrounded objects.

6 Conclusions and Future Work

We developed a model-based hatching approach for anatomical surfaces, in particular for vascular and elongated muscle structures derived from clinical volume datasets. Our novel combination of curvature and preferential direction information depicts local shape characteristics and supports structure distinctions. Avoiding time-consuming user interaction, providing real time capability, and frame coherence makes our approach applicable for intervention planning or visualizations in anatomical education. Additional shape perception is supported in contrast to a simple color shading, especially for semi-transparent context structures. For future work we intend to determine preferential directions for more anatomic or complex structures, e. g., bones or other muscle types.

Acknowledgment. We thank MeVis Research for providing advanced MeVisLab features.

References

1. Kim, S., Hagh-Shenas, H., Interrante, V.: Showing Shape with Texture: Two Directions Seem Better than One. In: SPIE Human Vision and Electronic Imaging, pp. 332–339 (2003)
2. Hertzmann, A., Zorin, D.: Illustrating Smooth Surfaces. In: SIGGRAPH, pp. 517–526 (2000)
3. Rössl, C., Kobbelt, L.: Line-Art Rendering of 3D-Models. Pacific Graphics, 87–96 (2000)
4. Ritter, F., Hansen, C., Dicken, V., Konrad-Verse, O., Preim, B., Peitgen, H.O.: Real-Time Illustration of Vascular Structures for Surgery. IEEE Transactions on Visualization 12, 877–884 (2006)
5. Rössl, C., Kobbelt, L., Seidel, H.P.: Line Art Rendering of Triangulated Surfaces using Discrete Lines of Curvature. In: WSCG, pp. 168–175 (2000)
6. Sousa, M.C., Samavati, F.F., Brunn, M.: Depicting Shape Features with Directional Strokes and Spotlighting. In: Computer Graphics International, pp. 214–221 (2004)
7. Zander, J., Isenberg, T., Schlechtweg, S., Strothotte, T.: High Quality Hatching. Computer Graphics Forum 23(3), 421–430 (2004)
8. Deussen, O., Hamel, J., Raab, A., Schlechtweg, S., Strothotte, T.: An Illustration Technique Using Hardware-Based Intersections. In: Proc. of Graphics Interface, pp. 175–182 (1999)
9. Praun, E., Finkelstein, A., Hoppe, H.: Lapped Textures. In: SIGGRAPH, pp. 465–470 (2000)
10. Praun, E., Hoppe, H., Webb, M., Finkelstein, A.: Real-Time Hatching. In: SIGGRAPH, pp. 581–586 (2001)
11. Gasteiger, R., Tietjen, C., Baer, A., Preim, B.: Curvature- and model-based surface hatching of anatomical structures derived from medical volume datasets. Technical report, University of Magdeburg, Department of Simulation and Graphics (2008)
12. Goldfeather, J.: Understanding Errors in Approximating Principal Direction Vectors. Technical report, University of Minnesota, Department of Computer Science and Engineering (2003)
13. Selle, D., Spindler, W., Preim, B., Peitgen, H.O.: Mathematical Methods in Medical Image Processing: Analysis of Vascular Structures for Preoperative Planning in Liver Surgery, pp. 1039–1059. Springer, Heidelberg (2000)

Relational Transparency Model for Interactive Technical Illustration

Ladislav Čmolík

Czech Technical University in Prague, Computer Graphics Group,
Karlovo námněstí 13, 12135 Prague 2, Czech Republic
cmolikl@fel.cvut.cz
http://www.cgg.cvut.cz

Abstract. Ghosted views are technical illustrations that communicate not only shape, but also inner structure and spatial relations of the depicted objects. Modulation of opacity is utilized to communicate the inner structure, therefore the spatial relations remains undistorted. In this paper an approach that enables real-time rendering of 3D ghosted views is presented. The advantage over classical handcrafted ghosted views is that the user is able to interact (e.g. pan, zoom and rotate) with the 3D model.

Keywords: Interactive technical illustration, Ghosted views, Transparency, Real-time non-photorealistic rendering.

1 Introduction

The need for visual communication of structure of 3D models is rising with the increasing complexity of 3D models. Nowadays 3D models are so complex that it may be very time consuming and challenging cognitive activity to comprehend. Therefore there is a need for tools that communicate the structure of a 3D model and thus give better insight into the subject that is represented by the model.

Technical illustration is an artistic discipline focusing on creation of 2D images that communicate shape, inner structure and spatial relations of the depicted objects. One can find such drawings in manuals, science books or even in advertisements.

In technical illustration there are three main techniques that communicate the inner structure of the depicted object - cut-away views, exploded views, and ghosted views. The cut-away views [3], [8] reveal important interior parts by cutting away the occluding exterior parts. The exploded views [9] rearrange positions of all parts in reverse of the assembly process rather than removing occluding geometry. The ghosted views reveal important interior parts by modulating opacity of exterior parts in areas where they occlude the interior parts (see Figure 1 as an example).

In this work we are focusing on interactive computer generated ghosted views of technical models. In [5] and [6] Dooley and Cohen presented an approach that allows to render static 2D ghosted views of 3D technical models. Diepstraten et al.

A. Butz et al. (Eds.): SG 2008, LNCS 5166, pp. 263–270, 2008.

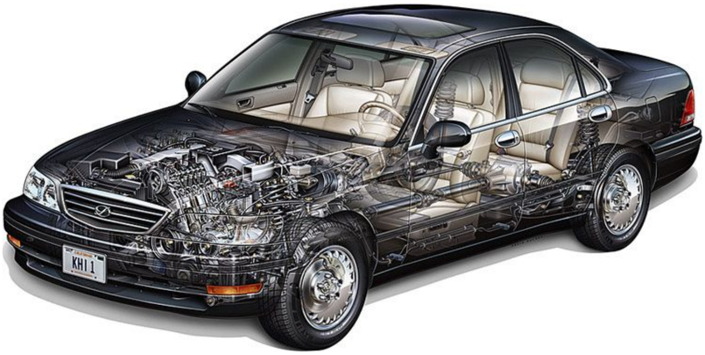


Fig. 1. An example of handcrafted ghosted view of a car. Image courtesy of Kevin Hulsey Illustration, Inc. Copyright ©2004 Kevin Hulsey Illustration, Inc. All rights reserved. Source [1].

have presented in [4] an approach focusing on interactive real-time rendering of ghosted views of technical models. The approach allows to render ghosted views with little user effort, but allows to modulate the opacity of the outermost geometry only. There are also several works [1], [14] dealing with ghosted views of volumetric datasets. However, they are not suitable for technical illustration due to nature of volumetric data that is very different from technical models.

In this work a novel multi-pass rendering technique that extends the approach of Diepstraten et. al. [4] and allows rendering of ghosted views at interactive frame rates is presented. In our approach not only the opacity of the outermost geometry but also the opacity of the nested geometry can be modulated. Thus hierarchical nesting of objects can be illustrated.

2 Principles Behind Ghosted Views

From handcrafted ghosted views created by artists [7] we have observed the following principles:

Grouping. Parts of the illustrated object are divided into several groups. The groups typically have some semantic meaning (e.g. interior of the car, chassis with engine - see Figure 1).

Layering. Each group is painted into separate layer independently of other groups.

Layer ordering. Group g_1 contained whole (or partially) inside a part from another group g_2 is painted into lower layer than group g_2.

Opacity modulation. Opacity of occluding parts is modulated in areas where important parts on lower layers are occluded.

Composition. The layers are composed in bottom-up order into the final illustration.

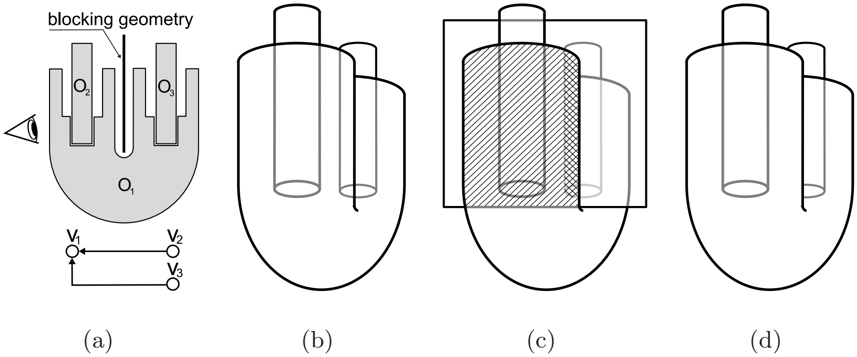


Fig. 2. (a) An example of more complicated arrangement of objects and graph G for this arrangement. (b) Illustration created using graph G only. (c) Blocking geometry associated to object O_1. The hatched area where object O_1 is before the blocking geometry is influenced. The opacity is not modulated in the crosshatched area where object O_3 is behind the blocking geometry. (d) Illustration created using graph G and the blocking geometry.

3 Relational Transparency Model

Our approach tries to follow the principles described in the previous section. Two user roles (illustrator and viewer) are considered in our approach. The illustrator annotates the 3D model with additional information and thus specifies the desired appearance of the illustrated 3D model. The viewer is then able to zoom, pan and rotate the model in real-time while the information specified by the illustrator influences the rendering process.

The approach is based on formal representation of specific relations between the objects. For purposes of the interactive technical illustration we introduce binary relation *show through* (denoted as $\rightarrow_s$). Let us consider objects O_1 and O_2 then the expression $O_1 \rightarrow_s O_2$ represents fact that object O_1 should show through object O_2 and thus the visibility of object O_1 can be maintained by modulating the opacity of object O_2 in the areas where it occludes object O_1.

The *show through* relation is irreflexive ($O_1 \not\rightarrow_s O_1$), asymmetric (if $O_1 \rightarrow_s O_2$ then $O_2 \not\rightarrow_s O_1$) and transitive (if $O_1 \rightarrow_s O_2$ and $O_2 \rightarrow_s O_3$ then $O_1 \rightarrow_s O_3$).

Let us consider a 3D model M that consists of n 3D objects $O_i \in M$, $i = 1 \ldots n$. Then the *show through* relations for the 3D model M are defined by oriented graph $G = (V, E)$. Each node $v_i \in V, i = 1 \ldots n$ represents 3D object O_i and an edge $e \in E$ from node v_k to node v_l represents the *show through* relation $O_k \rightarrow_s O_l$.

In certain cases when more complicated shapes of objects are considered the information contained in the graph G is not satisfactory. The arrangement of objects on Figure 2(a) illustrated as viewed from position indicated by the eye results into illustration on Figure 2(b). However, such arrangement of objects is typically illustrated by artists as depicted on Figure 2(d).

The problem can be solved with addition of blocking geometry associated to a specific object (see Figure 2(c)). In [4] Diepstraten et. al. promoted to use second front faces (acquired with depth peeling [10]) of the occluding objects (e.g. object O_1 on Figure 2) as the blocking geometry. However, for some arrangements of objects this approach results into incorrect illustration. Therefore we leave the creation and placement of blocking geometry to the illustrator. If the illustrator decides to use the occluding object itself as the blocking geometry then our approach results into the same effect as the approach of Diepstraten et. al. [4]. Fortunately, the blocking geometry is needed only for a small fraction of models.

The graph G in combination with the blocking geometry allows the illustrator to illustrate most of arrangements of 3D objects and still be able to control the desired appearance of the interactive illustration.

3.1 Rendering Order

The graph G allows us not only to determine which objects should show through another one, but also to determine the number of layers into which the objects should be painted and the order in which the layers should be composed together.

For irreflexive, asymmetric, and transitive relations a strict partial order can be found if and only if the graph G is directed acyclic graph (DAG) - see figure 3(a) as an example. Then the strict partial order is simply the transitive closure of G and it is also DAG.

The strict partial order divides the nodes of graph G into several groups. Each group characterizes that there is not an edge between two nodes in the group (see figure 3(b)). That is, that visibility of 3D objects represented by nodes in one group can be solved together and the objects can be rendered together into one layer.

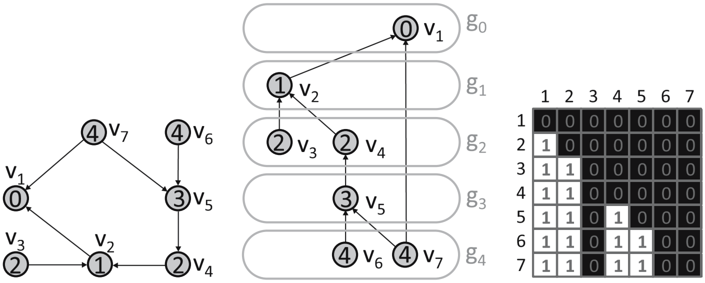
For each node $v_i, i \in 1 \ldots n$ of the transitive reduction of the graph G the length of all paths to other vertices is calculated and the length $l_i \in (0, n)$ of the longest path is stored. The number of layers needed is determined from the maximum of the lengths stored in all vertices $L = \max(l_1, l_2, \ldots l_n)$. The objects need to be rendered into $L+1$ layers to illustrate the 3D model correctly. Groups $g_0, g_1, \ldots g_L$ are initialized and the length l_i stored for each node v_i determines that object O_i belongs into group g_{l_i}.

Then the determination of the rendering order is straightforward. Note that group g_i contains only objects that show through objects in group g_j where $j < i$ (see Figure 3(b)). Therefore the groups have to be processed by GPU in descending order. Thus the back-to-front composition of the layers is ensured and the transparency is rendered correctly.

3.2 Rendering

Our rendering technique consists of two stages; preparation stage and execution stage. The preparation stage is performed for each 3D model only once. The execution stage is performed for each rendered frame.

In the preparation stage a 2D texture representing the node-node matrix of transitive closure of the given graph G is created. The node-node matrix can be



	1	2	3	4	5	6	7
1	0	0	0	0	0	0	0
2	1	0	0	0	0	0	0
3	1	1	0	0	0	0	0
4	1	1	0	0	0	0	0
5	1	1	0	1	0	0	0
6	1	1	0	1	1	0	0
7	1	1	0	1	1	0	0

(a) (b) (c)

Fig. 3. (a) An example of directed acyclic graph. The number in each node is length of the longest path of all paths to other nodes. (b) The groups the objects are divided into. (c) 2D texture representing the node-node matrix of the directed acyclic graph.

encoded into 2D RGB texture of the same size. Zeros can be represented with black color and ones with white color (see Figure 3(c)).

The implementation of the execution stage is based on G-buffers [13]. The technique is implemented as an OpenGL [12] multi-pass rendering algorithm. Frame buffers represent the layers into which the groups of objects are rendered. Color buffer, normal buffer, depth buffer, id buffer and buffer of average depth are associated to each frame buffer. Color buffer and normal buffer are represented with 32 bit floating point RGB textures. Depth buffer, id buffer and buffer of average depth are represented as red, green and blue component of one 32 bit floating point RGB texture. Note that the id of objects is used also to access the texture representing the node-node matrix of graph G.

First the groups g_L and g_{L-1} are rendered into frame buffers fb_1 and fb_2. Let us denote a picture element of frame buffer fb_1 as $p_1 = (x_1, y_1, c_1, n_1, d_1, id_1, ad_1 = d_1)$ where x_1 and y_1 are coordinates of the picture element in the frame buffer, c_1 is RGB color vector, n_1 is normal vector, d_1 is depth, id_1 is number of the object, and ad_1 is the average depth of object rendered into the pixel element. A picture element of frame buffer fb_2 is similarly denoted as $p_2 = (x_2, y_2, c_2, n_2, d_2, id_2, ad_2 = d_2)$.

Next the depth buffer of blocking geometry associated with objects in group g_{L-1} is rendered into frame buffer fb_3. A picture element of frame buffer fb_3 is denoted as $p_3 = (x_3, y_3, d_3)$ where x_3 and y_3 are coordinates and d_3 is depth. Only the blocking geometry associated with object its id is equal to id_2 on position (x_3, y_3) in frame buffer fb_2 and its depth is greater than d_2 on position (x_3, y_3) in frame buffer fb_2 is rendered onto position (x_3, y_3) in frame buffer fb_3.

Then the picture elements p_1, p_2 and p_3 on corresponding positions ($x_1 = x_2 = x_3$ and $y_1 = y_2 = y_3$) are composed into picture element p_4 of frame buffer fb_4 according to the following transparency rules:

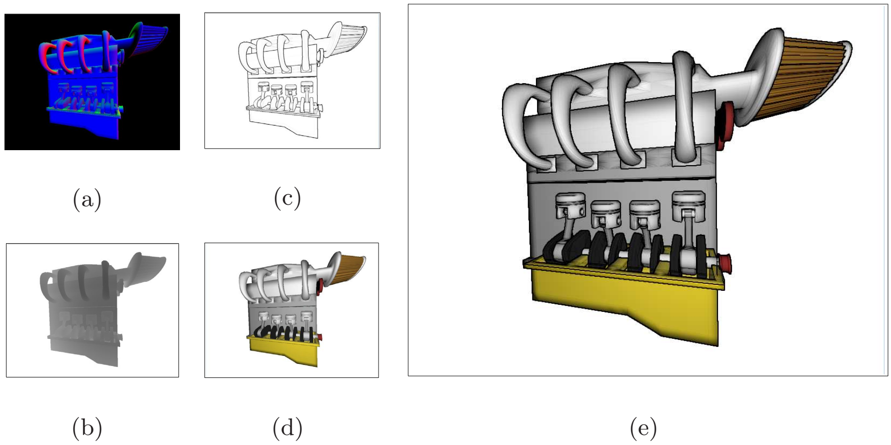
1. If $d_1 < d_2$ then picture element $p_4 = (x_1, y_1, c_1, n_1, d_1, id_1, ad_1)$ is created in frame buffer fb_4.

2. Otherwise
   (a) If the *show through* relation exists between objects that have been rendered into picture elements p_1 and p_2 exists and $d_1 < d_3$ then the opacity of the picture element p_2 is modulated. Therefore picture element $p_4 = (x_1, y_1, 0.8 \cdot c_1 + 0.2 \cdot c_2, 0.5 \cdot n_1 + 0.5 \cdot n_2, d_2, id_2, 0.5 \cdot ad_1 + 0.5 \cdot ad_2)$ is created in frame buffer fb_4. Note that the normal n_4 is average of n_1 and n_2. Similarly the average depth ad_4 is average of ad_1 and ad_2.
   (b) Otherwise pixel element $p_4 = (x_2, y_2, c_2, n_2, d_2, id_2, ad_2)$ is created in frame buffer fb_4.

Frame buffer fb_4 contains ghosted view of groups g_L and g_{L-1}. Then frame buffer fb_1 is renamed to fb_4 and vice versa. The next group g_{L-2} is rendered to frame buffer fb_2 and the blocking geometry to frame buffer fb_3. Then the frame buffers fb_1, fb_2 and fb_3 can be composed according to the same transparency rules. Similarly all subsequent groups $g_{L-3} \dots g_0$ are composed with the ghosted view created in the previous pass.

In the last step of the execution stage the silhouettes and crease edges of the ghosted view are detected. The silhouettes are detected as discontinuities in the buffer of average depths [11] and the crease lines are detected as discontinuities in the normal buffer [2].

Note that the depths and the normals are average depths and average normals of all composed groups (see transparency rule 2a in previous section). This allows us to detect silhouettes and crease lines of all combined layers at once. As an example see Figure 4.



(a) (c)

(b) (d) (e)

Fig. 4. (a) Normal buffer containing the average normals of all composed layers. (b) The buffer of average depths of all composed layers. (c) The silhouettes and crease lines detected as discontinuities in the normal buffer and in the buffer of average depths. (d) The color buffer of all composed layers. (e) The composition of the silhouettes and crease lines with the color buffer.

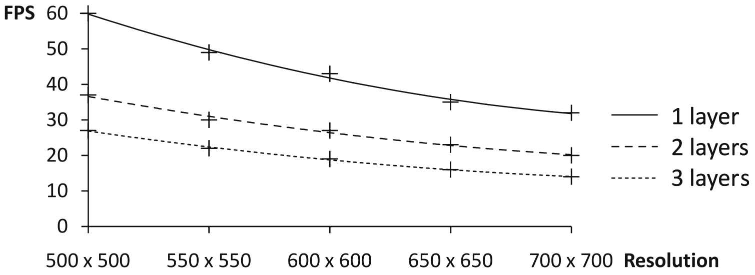


Fig. 5. Performance of our algorithm in dependence on number of composed layers and resolution

3.3 Performance

The performance was measured on computer equipped with Core 2 Duo 1.5 GHz processor, 2 GB of RAM, and NVIDIA Quadro NVS 140M with 128 MB of RAM and 16 stream units. On this configuration we have achieved 27 fps for 3D model of engine (see Figure 4(e)) containing 65 objects (103 046 faces) that was divided into 3 groups. The model was rendered in resolution 500 × 500.

The algorithm scales very good with increasing number of faces, because each 3D object is rendered only once. The only overhead is the blocking geometry. How the algorithm scales with resolution, and with number of composed layers is shown on Figure 5. The performance of the algorithm depends significantly on the number of stream units of the GPU, because of the parallel processing of vertices and fragments. On the modern GPUs we can expect significant performance increase, because the modern GPUs have up to 320 stream units.

4 Conclusion

An approach that enables illustrators to create interactive ghosted views of 3D models was introduced. The illustrator annotates 3D model with the show through relations and with the blocking geometry and thus specifies the desired appearance of the interactive illustration. Then the viewer can interact (pan, zoom and rotate) with the illustrated 3D model in real-time while the communication of structure, shape and spatial properties of the 3D model is preserved.

In our approach the illustrator has much better control over the final appearance of the interactive illustration than in case when previous approaches are considered. Our approach does not violate principles that are used by illustrators to communicate inner structure in 2D handcrafted technical illustration.

Acknowledgement

This research has been supported by the MSMT under the research program MSM 6840770014 and by the MSMT under the research program LC-06008 (Center for Computer Graphics).

References

1. Bruckner, S., Grimm, S., Kanitsar, A., Groller, M.E.: Illustrative context-preserving volume rendering. In: EUROVIS 2005: Eurographics / IEEE VGTC Symposium on Visualization, Aire-la-Ville, Switzerland, pp. 69–76 (2005)
2. Decaudin, P.: Cartoon-looking rendering of 3D-scenes. Tech. Rep. INRIA 2919, Universite de Technologie de Compiegne, France (1996)
3. Diepstraten, J., Weiskopf, D., Ertl, T.: Interactive cutaway illustrations. Computer Graphics Forum 22, 523–532 (2003)
4. Diepstraten, J., Weiskopf, D., Ertl, T.: Transparency in Interactive Technical Illustrations. Computer Graphics Forum 21, 317–326 (2002)
5. Dooley, D., Cohen, M.F.: Automatic Illustration of 3d Geometric Models: Lines. In: SI3D 1990: Proceedings of the 1990 symposium on Interactive 3D graphics, pp. 77–82. ACM Press, New York (1990)
6. Dooley, D., Cohen, M.F.: Automatic Illustration of 3d Geometric Models: Surfaces. In: VIS 1990: Proceedings of the 1st conference on Visualization 1990, pp. 307–314. IEEE Press, Los Alamitos (1990)
7. Kevin Hulsey Ilustration, Inc.: Technical Illustration by Kevin Hulsey Illustration, Inc., `http://www.khulsey.com/main_english.html`
8. Li, W., Ritter, L., Agrawala, M., Curless, B., Salesin, D.H.: Interactive cutaway illustrations of complex 3D models. ACM Transactions on Graphics (Proc. SIGGRAPH) 26(3), 31:1–31:11 (2007)
9. Li, W., Agrawala, M., Salesin, D.H.: Interactive image-based exploded view diagrams. In: Graphics Interface 2004, pp. 203–212 (2004)
10. Mammen, A.: Transparency and Antialiasing Algorithms Implemented with the Virtual Pixel Maps Technique. IEEE Comput. Graph. Appl. 9(4), 43–55 (1989)
11. Saito, T., Takahashi, T.: Comprehensible rendering of 3-d shapes. SIGGRAPH Comput. Graph. 24(4), 197–206 (1990)
12. Shreiner, D., Woo, M., Neider, J., Davis, T.: OpenGL Programming Guide: The Official Guide to Learning OpenGL, Version 2.1, 6th edn. Addison-Wesley, Upper Saddle River (2007)
13. Strothe, T., Schlechtweg, S.: Non-Photorealistic Computer Graphics: Modeling, Rendering and Animation. Morgan Kaufmann, San Francisco (2002)
14. Viola, I., Kanitsar, A.: Importance-driven feature enhancement in volume visualization. IEEE Transactions on Visualization and Computer Graphics 11(4), 408–418 (2005)

An Interactive Large Graph Visualizer

Hiroshi Hosobe

National Institute of Informatics
2-1-2 Hitotsubashi, Chiyoda-ku, Tokyo 101-8430, Japan
hosobe@nii.ac.jp

1 Introduction

Real-world information can often be expressed as a *graph* that consists of *nodes* and *edges*. Graphs are often effectively represented in a visual manner. To make visual representations of graphs easily available, researchers have proposed numerous techniques for automatic *visualization* of graphs [1].

An important class of graphs is the *general undirected graph*, which imposes no restrictions on its structure as well as having no directed edges. The force-directed approach [1] is often adopted to visualize general undirected graphs, and works well for graphs with hundreds of nodes.

However, it is still difficult to visualize complex general undirected graphs with over thousands of nodes. Examples of such graphs include social networks and computer networks, and there is a growing demand for visualizing such graphs. Unlike planar graphs and mesh-structured graphs, such graphs inevitably result in layouts with many crossing edges.

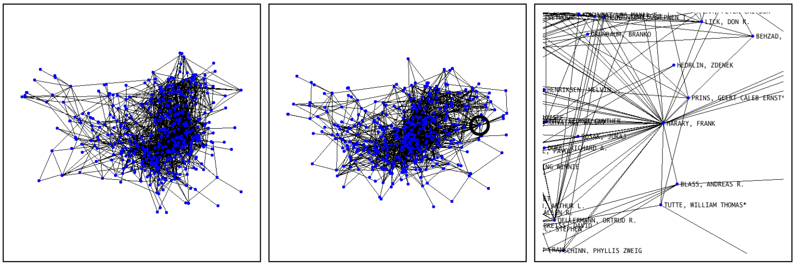
This demonstration provides a system for visualizing large complex general undirected graphs. We adopt *interactive visualization* of graphs, which allows users to focus on interesting parts of graphs. Our research goal is to enable efficient dynamic re-layout of graphs rather than static aesthetic layout of graphs. The rest of this manuscript presents our *high-dimensional approach* [2] and the *AGI graph visualization system* [3], also describing our future directions.

2 High-Dimensional Approach

The high-dimensional approach aims at interactive visualization of large graphs. The key idea is to divide the graph visualization process into two phases: (1) the *preprocessing phase* obtains a static graph layout in a very high-dimensional space; (2) the *interaction phase* dynamically computes the 2D graph layout by projecting the high-dimensional graph layout onto an appropriately determined 2D plane. This separation enables efficient interactive graph visualization. The high-dimensional graph layout needs to be computed only once by the preprocessing phase. After that, the user's interaction for updating the 2D graph layout is fulfilled by repeatedly executing the interaction phase, which is completed in a much shorter time than the preprocessing phase.

Our first method [2] based on this approach uses an extended version of Kruskal and Seery's method [4] to obtain high-dimensional graph layouts. It

A. Butz et al. (Eds.): SG 2008, LNCS 5166, pp. 271–272, 2008.



(a) (b) (c)

Fig. 1. Interactive visualization of a social network using AGI

exploits constraint programming to determine appropriate 2D planes onto which high-dimensional layouts will be projected.

3 AGI System

We are developing an interactive graph visualization system called AGI by using the high-dimensional approach. This system is written in C++, and can be executed on multiple platforms including Windows, Mac OS X, and Linux.

Figure 1 shows an interactive visualization of a graph using AGI. This graph represents a social network of mathematicians provided by the Erdös Number Project.[1] In this example, after an initial graph layout is obtained (Fig. 1(a)), the user drags a node of interest to a sparser area (Fig. 1(b)), and zooms in around the dragged node (Fig. 1(c)). This graph consists of 463 nodes and 1,547 edges, and each execution of the interaction phase is completed within 10 milliseconds.

4 Future Directions

We are still enhancing the AGI system. One of our future directions is to improve its performance and usability by developing a more sophisticated interactive graph visualization method based on the high-dimensional approach. Other directions include interactive 3D graph visualization based on this approach.

References

1. Di Battista, G., Eades, P., Tamassia, R., Tollis, I.G.: Graph Drawing: Algorithms for the Visualization of Graphs. Prentice Hall, Upper Saddle River (1999)
2. Hosobe, H.: A high-dimensional approach to interactive graph visualization. In: Proc. ACM SAC, pp. 1253–1257 (2004)
3. Hosobe, H.: An extended high-dimensional method for interactive graph drawing. In: Proc. APVIS, CRPIT, vol. 45, pp. 11–16. Aust. Comput. Soc. (2005)
4. Kruskal, J.B., Seery, J.B.: Designing network diagrams. In: Proc. General Conf. on Social Graphics, pp. 22–50. U.S. Dept. Census (1980)

[1] http://www.oakland.edu/enp/

Panel Beat: Layout and Timing of Comic Panels

William Bares

Millsaps College, Department of Computer Science,
Jackson MS 39210, USA
{bareswh}@millsaps.edu
http://home.millsaps.edu/bareswh/

Abstract. In comic book conventions, the size of a panel is generally proportional to the amount of narrative time that passes or the amount of time a reader spends viewing the panel. This poster paper describes work in progress to develop an automated system to assist in generating comic book panel layouts from a given sequence of panel artwork, the desired narrative duration of each panel, desired reading order, and a user-provided library of preferred page layout style guidelines.

1 Introduction

This poster paper describes work in progress to implement an automated interactive assistant for arranging and sizing comic book panels to follow conventions of narrative duration and reading order. In comic conventions, larger panels tend to hold the reader's attention for a longer time. Larger panels can also suggest a longer duration of elapsed narrative time or a slower pace of narrative action [1, 5]. To use this automated comic book layout assistant, an artist inputs a sequence of images for the artwork and balloons in each panel. Some artists, such as Rivka (alias), designate the pacing of a page by a *timing string* for example, "DBBAABB." The duration of each panel ranges from 'A' to 'E', with 'A' being the shortest and 'E' being the longest. The artist provides a library of pre-built page layout templates which represent desirable layouts and pacing patterns. The artist also specifies the reading order as left-to-right or right-to-left. For western readers, panels are read moving left-to-right then top-to-bottom. Japanese comics are read moving right-to-left then top-to-bottom. Given these inputs, the automated assistant sizes each panel to correspond to its respective duration and places the panels, whenever possible, to conform to the specified layout templates having similar timing strings. The artist can then flip through the pages of the comic to inspect the layout. The artist can modify the duration of one or more panels and invoke the assistant to re-compute the layout.

2 Related Work

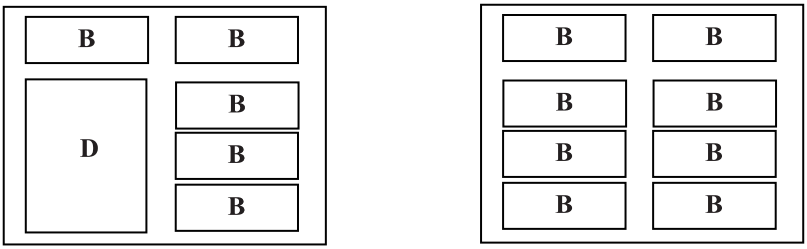
Recent works in automated generation of comics have summarized digital video [7], online chat transcripts [3], and events in virtual worlds [6]. Lok and Feiner summarize constraint- and learning-based automated layout systems [4]. This algorithm utilizes

A. Butz et al. (Eds.): SG 2008, LNCS 5166, pp. 273–276, 2008.

implied low-level constraints of panel size, fit within page, and non overlap of panels. Jacobs et al compute layouts of text and figures using dynamically adaptable grid layout templates to adapt to different display devices [2]. This work shares the idea of using flexible templates, but addresses the sub-problem of comics. Uchihashi et al. size keyframe panels by video duration and pack panels into a grid using an exhaustive row-wise search, but their work is not directed at producing layouts directed to specified style guides [7].

3 Panel Layout Library

The artist begins by using the layout editor to construct a library of page layout templates to provide examples of his or her style (Figure 1a). The set of templates is organized into an index structure that will be used to retrieve layout templates that most closely match the desired timing string of the input sequence of artwork panels.



(a) Layout template for "BBDBBB"

(b) Derived template for "BBBBBBBB"

Fig. 1. Example of layout templates

The list of rectangular panels for a page layout template is automatically converted into a recursive structure called a `GuidePanelLayout(GPL)` that contains a list of elements. Each element is either a panel or another `GuidePanelLayout`. Each GPL specifies its orientation (vertical or horizontal), its bounding rectangle, and its *timing string*. Assuming a left-to-right reading order, the timing string for the panels shown in Figure 1a is "BBDBBB." The recursive GPL structure for the above panels is:

```
GPL-Vertical {   GPL-Horizontal { B, B },
                 GPL-Horizontal { GPL-Vertical { D }
                                  GPL-Vertical { B, B, B } } }
```

The assistant automatically recognizes child elements, e.g., GPL-Vertical { B, B, B }, as first-class GPL's enabling it to derive additional templates for the library (Figure 1b).

4 Page Layout Algorithm

The automated layout assistant first tries to find a page layout template or a child element whose timing string exactly matches the timing pattern of the next unprocessed timing substring of the given input panels. Whenever a match is found, the sequence of input panels adopts the layout specified by the page layout template or child element. Otherwise, the automated layout assistant applies a recursive backtracking constraint-satisfaction problem (CSP) search to find a workable layout. The constraint problem is formulated such that there is a variable to determine how each panel is arranged relative to its immediate predecessor in the specified reading order. For a left-to-right reading order, the next panel may be stationed 'horizontally' that is to the right of and as far up as possible (without hitting another panel or the edge of the page) from the previous panel. Or, the next panel may be positioned 'vertically' that is below and as far to the left as possible from the previous panel. Hence, the domain of each variable consists of the values 'horizontal' or 'vertical'. Panel placements that would fall outside of the page are rejected. Backtrack if neither 'horizontal' nor 'vertical' placements are possible. If both placements are possible, then proceed with the one that leaves the least amount of empty space in the bounding rectangle of the panels placed so far. Once no more panels can be placed on a page, then output the set of panels for one page and repeat the process.

5 Implementation

The automated layout assistant and page layout template editor are coded in Java. In one test case, a sequence of 224 panels repeats a timing string of "DBBAABB." The layout for these 224 panels was formatted in 32 pages and was computed in 26 milliseconds (10.4 seconds over 400 repetitions) on a 2 GHz Intel Core2 Duo iMac running Mac OS X 10.4.1. A screen capture of the panel layout for one of these pages is shown in Figure 2.

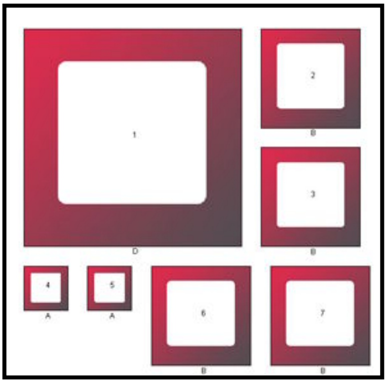


Fig. 2. Computed panel layout for one of the 32 pages in the test case

6 Conclusions and Future Work

The assistant cam compute multi-page layouts for hundreds of panels at interactive rates. More work is needed to automatically generate GPL's in the style of a small number of artist-specified samples. The algorithm too often defaults to backtracking when a given timing string does not match a template. For example, variations in panel aspect ratio make it difficult to find matches. We plan more work to better align panel edges over the whole page, adjust panel sizes to utilize empty page space, and apply stylistic variations such as partially overlapping panels. We plan to solicit hands-on feedback from comic layout artists to improve the usability of the interactive layout assistant.

Acknowledgements

Thanks to Millsaps Computer Science majors Franklin Landry and Tyler Moss. Franklin designed a set of layout templates inspired by manga and is completing a comic editor application. Tyler contributed Java code to implement backtracking CSP search.

References

1. Eisner, W.: Comics and Sequential Art. Poorhouse Press, Paramus (1985)
2. Jacobs, C., Li, W., Schrier, E., Bargeron, D., Salesin, D.: Adaptive Grid-Based Document Layout. ACM Transactions on Graphics 22(3); Proceedings of ACM SIGGRAPH 2003, July 2003, pp. 838–847 (2003)
3. Kurlander, D., Skelly, T., Salesin, D.: Comic Chat. In: Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques SIGGRAPH 1996, New Orleans, LA, August 4-9, 1996, pp. 225–236 (1996)
4. Lok, S., Feiner, S.: A Survey of Automated Layout Techniques for Information Presentations. In: Proceedings of Smart Graphics 2001, Hawthorne NY, pp. 61–68 (2001)
5. Rivkah.: Paneling, Pacing, and Layout in Comics and Manga, `http://lilrivkah.livejournal.com/168859.html`
6. Shamir, A., Rubinstein, M., Levinboim, T.: Generating Comics from 3D Interactive Computer Graphics. IEEE Computer Graphics & Applications 26(3), 53–61 (2006)
7. Uchihashi, S., Foote, J., Girgensohn, A., Boreczky, J.: Video Manga: Generating Semantically Meaningful Video Summaries. In: Proceedings of ACM Multimedia, pp. 383–392 (1999)

Author Index