Studies in Computational Intelligence 213

Stefano Cagnoni (Ed.)

Evolutionary Image Analysis and Signal Processing

Springer

Stefano Cagnoni (Ed.)

Evolutionary Image Analysis and Signal Processing

Studies in Computational Intelligence, Volume 213

Editor-in-Chief
Prof. Janusz Kacprzyk
Systems Research Institute
Polish Academy of Sciences
ul. Newelska 6
01-447 Warsaw
Poland
E-mail: kacprzyk@ibspan.waw.pl

Stefano Cagnoni (Ed.)

Evolutionary Image Analysis and Signal Processing

Springer

Dr. Stefano Cagnoni
Dipartimnto di Ingegneria dell'Informazione
Viale Usberti 181a
43100 PARMA
Italy
E-mail: cagnoni@ce.unipr.it

ISBN 978-3-642-01635-6 e-ISBN 978-3-642-01636-3

DOI 10.1007/978-3-642-01636-3

Studies in Computational Intelligence ISSN 1860949X

Library of Congress Control Number: Applied for

Typeset & *Cover Design:* Scientific Publishing Services Pvt. Ltd., Chennai, India.

Printed in acid-free paper

9 8 7 6 5 4 3 2 1

springer.com

Preface

The publication of this book on evolutionary Image Analysis and Signal Processing (IASP) has two main goals. The first, occasional one is to celebrate the 10th edition of EvoIASP, the workshop which has been the only event specifically dedicated to this topic since 1999. The second, more important one is to give an overview of the opportunities offered by Evolutionary Computation (EC) techniques to computer vision, pattern recognition, and image and signal processing.

It is not possible to celebrate EvoIASP properly without first acknowledging EvoNET, the EU-funded network of excellence, which has made it possible for Europe to build a strong European research community on EC. Thanks to the success of the first, pioneering event organized by EvoNET, held in 1998 in Paris, it was possible to realize that not only was EC a fertile ground for basic research but also there were several application fields to which EC techniques could offer a valuable contribution. That was how the idea of creating a single event, EvoWorkshops, out of a collection of workshops dedicated to applications of EC, was born. Amongst the possible application fields for EC, IASP was selected almost accidentally, due to the occasional presence, within EvoNET, of less than a handful of researchers who were interested in it. I would lie if I stated that the event was a great success since its very start, but it was successful enough to survive healthily for a couple of years, before reaching its present size, relevance, and popularity.

The papers selected for inclusion in this book, mostly extended versions of papers presented at EvoIASP, have no pretence of setting milestones in the history of evolutionary IASP, but they do offer readers a panoramic view of what can be currently done using EC techniques in such applications. From this point of view, what could be seen as a defect of this book, at first sight, can even become a useful and peculiar feature. In fact, this rather unstructured collection of papers samples the space of Evolutionary IASP rather extensively, albeit possibly sparsely, along different axes. The most obvious one is related to applications in different areas of IASP, as the book describes a wide variety of applications in which

EC can fruitfully be employed. However, there are less obvious ones. Amongst these, let me mention and discuss what I believe is a very important one: the ‘degree of involvement’ of EC in IASP applications. The book describes applications in which EC techniques play very different roles in producing the final results. Differently from how EC is most commonly looked upon, or perceived, by non-EC researchers, EC techniques can actually represent more than an external optimization tool that can be used to tune or refine parameters or components of a mostly pre-defined solution. In fact, EC techniques can be embedded more intimately into IASP applications, up to situations where the solution itself is intrinsically evolutionary. This book provides examples that are positioned at both ends of this axis. Of course, that is not the only possible ordering criterion that can be used to create a taxonomy of Evolutionary IASP. In a recent paper, I mentioned at least two more criteria, perhaps the most natural ones out of many possible: EC-based, according to the evolutionary paradigm that is used, and application-based, according to the abstraction level of the task to which EC is applied.

Deciding the ordering criterion for the contributions in this book has been no easy task, as they were mostly extended versions of papers that have been presented at EvoIASP. Therefore, there is neither a pre-established logical structure underlying their choice, nor was it possible to find any ordering with respect to which they could appear to be uniformly distributed. Because of this, I decided not to subdivide the book into sections. Nevertheless, an application-based ordering criterion is implicitly followed, with some additional constraints which reflect the presence of more work dealing with topics and applications belonging to the computer vision domain. Contributions belonging to this larger group appear first, and are ordered according to the abstraction level of the task they describe. In the following smaller set of contributions, more general and basic tasks are tackled, which, with some extension of their context, can also find effective applications in computer vision. An implicit secondary EC-based indexing criterion has also been followed by trying to keep applications based on the same or similar EC paradigms close to one another.

From the point of view of the expected audience, even if the contents of this book are derived from a workshop that is addressed mainly to EC researchers, this book does not specifically address any category of readers. Knowledge of the most basic EC paradigms is given for granted, while, possibly, some more basic detail about the specific applications is given, which may be obvious to readers who are familiar with them. However, all authors have made their best efforts to keep their contributions as balanced as possible for the reading to be enjoyable for the widest possible audience. Any variations to the basic evolutionary algorithms or application-related functions are described in details. On the other hand, details about the specific applications are usually limited to the information that is essential for their understanding.

A common problem that occurs when techniques developed within a specific community are applied to a number of fields for which other well-established communities also exist is that members of each community tend to publicize their work within their 'natural environment'. The result is that, first, similar work is often described very differently, as some authors focus mainly on applications, while others concentrate on methodology. Second, and more important, a lack of communication occurs by which researchers belonging to one community tend to keep a very partial view of the topics pertaining to the other communities. As a result, on the one hand, researchers in the application fields tend to consider basic methods to be well-established, ready-for-use, closed tools; on the other hand, those who do basic research often tend to consider application-related data as abstract benchmarks using which their results can be tested, neglecting their actual meaning in the real world. One of the most appealing features of books like this is being, in general, more universally visible and less community-specific than, for example, conference proceedings, besides, obviously, having a much narrower scope than the latter. In its first 10 years, EvoIASP has, hopefully with success, sowed the seeds for a new 'multi-cultural' community of researchers in evolutionary IASP. I do wish this book will further contribute to the widening of this community, both numerically and in terms of research interests, and that we will celebrate more successes, and the 20th edition of the workshop, 10 years from now.

Parma,
January 2009 Stefano Cagnoni

Acknowledgements

This book is dedicated, in first place, to all those who made it possible for EvoIASP to exist and survive in good health for 10 years (which will likely be 11 when this book is published):

- Riccardo Poli, a friend and colleague, who introduced me to Evolutionary Computation and showed me the way into this fascinating field when we were still PhD students. Then, when he was already one of the most active and influential members of the EC community, it was him who proposed that I co-chaired the EvoIASP working group in the early years of EvoNET, the EU-funded Network of Excellence on Evolutionary Computation;
- Terry Fogarty, co-chair of the first editions of EvoIASP, but most of all a pioneer in the field of Evolutionary Computation, co-ordinator of EvoNET as well as a friendly and hilarious companion of pleasant after-conference chats;
- Jennifer Willies, EvoNET and Evo* coordinator, as indispensable as discrete support for all EvoNET events, gifted by an incredible ability to make the best out of the budgets she has to manage, often limited and severely constrained. Acknowledging only her professional achievements would be more than restrictive. Just ask any EvoNET member or participant in Evo* for more details on her availability, patience and motherly care in any of the (infinite) situations where her intervention is requested, and
- All those who have submitted and presented their work at EvoIASP, with particular regards to those who, after their first participation in the workshop, have been engaged in Evolutionary Image Analysis and Signal Processing and in the workshop itself. Amongst these, I would like to thank, in particular, Evelyne Lutton, Gustavo Olague, Jean Louchet and Mengjie Zhang, as well as all who contributed to the workshop as reviewers.

I would also like to thank very warmly all authors of the chapters included in this book for their patience in preparing excellent extensions of their work presented at EvoIASP, and especially for coping with my slowness in turning their contribution into the volume you are reading.

Contents

List of Contributors

Raed Abu Zitar
School of Computer Science and Engineering,
New York Institute of Technology,
Amman, Jordan
`rzitar@philadelphia.edu.jo`

Ayman Al-Dmour
Department of Information Technology, Al-Hussein Bin Talal University,
Maán, Jordan
`d.ayman@ahu.edu.jo`

Thomas Bäck
Natural Computing Group,
Leiden Institute of Advanced Computer Science,
Leiden University, The Netherlands
`baeck@liacs.nl`

Ernst G.P. Bovenkamp
Division of Image Processing,
Department of Radiology C2S,
Leiden University Medical Center,
The Netherlands
`E.G.P.Bovenkamp@lumc.nl`

Nareli Cruz-Cortés
Center for Computing Research,
National Polytechnic Institute,
Zacatenco 07738,
Mexico City, México
`nareli@cic.ipn.mx`

Ivanoe De Falco
ICAR–CNR, Via P. Castellino 111,
80131 Naples, Italy
`ivanoe.defalco@na.icar.cnr.it`

Luis G. De la Fraga
Cinvestav, Department of Computing, IPN 2508,
07360 Mexico City, México
`fraga@cs.cinvestav.mx`

Antonio Della Cioppa
Natural Computation Lab,
DIIIE, University of Salerno,
Via Ponte don Melillo 1,
84084 Fisciano (SA), Italy
`adellacioppa@unisa.it`

Jouke Dijkstra
Division of Image Processing,
Department of Radiology C2S,
Leiden University Medical Center,
The Netherlands
`J.Dijkstra@lumc.nl`

Jeroen Eggermont
Division of Image Processing,
Department of Radiology C2S,
Leiden University Medical Center,

The Netherlands
J.Eggermont@lumc.nl

Michael T.M. Emmerich
Natural Computing Group,
Leiden Institute of
Advanced Computer Science,
Leiden University, The Netherlands
emmerich@liacs.nl

Francisco Fernández
University of Extremadura,
Computer Science Department,
Centro Universitario de Mérida,
C/Sta Teresa de Jornet, 38,
06800 Mérida, Spain
fcofdez@unex.es

Kyrre Glette
University of Oslo,
Department of Informatics,
P.O. Box 1080 Blindern,
0316 Oslo, Norway
kyrrehg@ifi.uio.no

Wojciech Jaśkowski
Institute of Computing Science,
Poznan University of Technology,
Piotrowo 2, 60965 Poznań, Poland
wjaskowski@cs.put.poznan.pl

Mark Johnston
School of Enginering and
Computer Science,
Victoria University of Wellington,
P.O. Box 600,
Wellington, New Zealand
mark.johnston@vuw.ac.nz

Krzysztof Krawiec
Institute of Computing Science,
Poznan University of Technology,
Piotrowo 2, 60965 Poznań, Poland
kkrawiec@cs.put.poznan.pl

Kristijan Lenac
A.I.B.S. Lab S.r.l.,
Via del Follatoio 12, Trieste, Italy
klenac@units.it

Rui Li
Natural Computing Group,
Leiden Institute of
Advanced Computer Science,
Leiden University, The Netherlands
ruili@liacs.nl

Evelyne Lutton
INRIA Rocquencourt,
Complex Team,
Domaine de Voluceau BP 105,
78153 Le Chesnay Cedex, France
evelyne.lutton@inria.fr

Domenico Maisto
ICAR–CNR, Via P. Castellino 111,
80131 Naples, Italy
domenico.maisto@na.icar.cnr.it

Enzo Mumolo
DEEI, University of Trieste,
Via Valerio 10,
Trieste, Italy
mumolo@units.it

Ferrante Neri
Department of Mathematical
Information Technology, Agora,
University of Jyväskylä,
P.O. Box 35 (Agora),
FI-40014 Jyväskylä, Finland
neferran@cc.jyu.fi

Massimiliano Nolich
DEEI, University of Trieste,
Via Valerio 10,
Trieste, Italy
mnolich@units.it

Gustavo Olague
CICESE, Research Center,
División de Física Aplicada
Centro de Investigación
Científica y de Educación
Superior de Ensenada, B.C.,
Km. 107 Carretera

Tijuana-Ensenada,
22860, Ensenada, B.C., México
olague@cicese.mx

Cynthia B. Pérez
CICESE, Research Center,
División de Física Aplicada
Centro de Investigación
Científica y de Educación
Superior de Ensenada, B.C.,
Km. 107 Carretera
Tijuana-Ensenada,
22860, Ensenada, B.C., México
cbperez@cicese.mx

Johan H.C. Reiber
Division of Image Processing,
Department of Radiology C2S,
Leiden University Medical Center,
The Netherlands
Johan H.C. Reiber@lumc.nl

Umberto Scafuri
ICAR–CNR, Via P. Castellino 111,
80131 Naples, Italy
umberto.scafuri@na.icar.cnr.it

Ernesto Tarantino
ICAR–CNR, Via P. Castellino 111,
80131 Naples, Italy
ernesto.tarantino@
na.icar.cnr.it

Ville Tirronen
Department of Mathematical
Information Technology, Agora,
University of Jyväskylä,
P.O. Box 35 (Agora),
FI-40014 Jyväskylä, Finland
aleator@cc.jyu.fi

Jim Torresen
University of Oslo,
Department of Informatics,
P.O. Box 1080 Blindern,
0316 Oslo, Norway
jimtoer@ifi.uio.no

Leonardo Trujillo
CICESE, Research Center,
División de Física Aplicada
Centro de Investigación
Científica y de Educación
Superior de Ensenada, B.C.,
Km. 107 Carretera
Tijuana-Ensenada,
22860, Ensenada, B.C., México
trujillo@cicese.mx

Israel Vite Silva
Cinvestav,
Department of Computing,
Av. IPN 2508. 07360,
Mexico City, México
ivite@
computacion.cs.cinvestav.mx

Bartosz Wieloch
Institute of Computing Science,
Poznan University of Technology,
Piotrowo 2, 60965 Poznań, Poland
bwieloch@cs.put.poznan.pl

Moritoshi Yasunaga
University of Tsukuba,
Graduate School of Systems and
Information Engineering,
1-1-1 Ten-ou-dai,
Tsukuba, Ibaraki, Japan
yasunaga@cs.tsukuba.ac.jp

Mengjie Zhang
School of Enginering and
Computer Science,
Victoria University of Wellington,
P.O. Box 600,
Wellington, New Zealand
mengjie.zhang@ecs.vuw.ac.nz

Evolutionary Image Analysis and Signal Processing

Texture Image Segmentation Using an Interactive Evolutionary Approach

Cynthia B. Pérez, Gustavo Olague, Evelyne Lutton, and Francisco Fernández

Abstract. This work presents the Interactive Evolutionary Segmentation algorithm, *I-EvoSeg*, an extension of the EvoSeg algorithm [18]. The proposed approach uses texture information as image features and identifies regions automatically using a homogeneity criterion. Furthermore, *I-EvoSeg* complements the chosen texture information with direct human interaction in the evolutionary optimization process. Interactive evolution helps to improve results by allowing the algorithm to adapt using the new external information based on user evaluation. Texture information is extracted from the Grey Level Co-occurrence Matrix (GLCM) using statistical descriptors. The fitness evaluation of the *I-EvoSeg* algorithm is twofold: (a) Internal fitness provided by the local and global minimum distance between regions and (b) External fitness that depends on the expertise of the user who participates during the evaluation process. We tested *I-EvoSeg* using texture images and compared it with the standard EvoSeg algorithm. Experimental results show that the interactive approach produces qualitatively better segmentation.

1 Introduction

Human vision is a complex process that is not yet completely understood despite several decades of research from the standpoint of natural science and artificial

Cynthia B. Pérez and Gustavo Olague
CICESE, Research Center, División de Física Aplicada
Centro de Investigación Científica y de Educación Superior de Ensenada,
B.C., Km. 107 Carretera Tijuana-Ensenada, 22860, Ensenada, B.C., México
e-mail: cbperez@cicese.mx, olague@cicese.mx

Evelyne Lutton
INRIA Rocquencourt, Complex Team
Domaine de Voluceau BP 105
78153 Le Chesnay Cedex, France
e-mail: evelyne.lutton@inria.fr

Francisco Fernández
University of Extremadura, Computer Science Department
Centro Universitario de Merida, C/Sta Teresa de Jornet, 38. 06800 Mérida, Spain
e-mail: fcofdez@unex.es

intelligence. For example, the detection of a camouflaged chameleon in the desert is a perceptual task that can be achieved by some human observer; however, this task is more difficult for a computer system. The cause of failure is the difficulty in separating the reptile from the environmental background, a task that requires different types of information, e.g. surface brighteness, shape, colour, texture and movements. In computer vision, the complex cognitive process of identifying colours, shapes, textures and automatically grouping them into separate objects within a scene is called image segmentation, which is still an open line of research despite years of formal inquiry.

Image segmentation is a process in which an input image is partitioned into regions that are homogeneous according to some group of characteristics, e.g. texture information. Formally, image segmentation could be defined as follows:

Definition 1. Segmentation of I is a partition P of I into a set of M regions R_m, $m = 1,2,...,M$, such that:

$$\begin{aligned}
&1)\ \bigcup_{m=1}^{M} R_m = I \quad \text{with} \quad R_m \bigcap R_n = \emptyset, \quad m \neq n \\
&2)\ H(R_m) = true \quad \forall\ m \\
&3)\ H(R_m \bigcup R_n) = false \quad \forall\ R_m \ \text{and}\ R_n \ \text{adjacent}
\end{aligned} \tag{1}$$

where I is the image and H is the predicate of homogeneity. Thus, each region in a segmented image needs to simultaneously satisfy the properties of homogeneity and connectivity [1]. A region is homogeneous if all of its pixels satisfy a homogeneity predicate defined over one or more pixel attributes such as intensity, texture or colour. Moreover, a region is said to be connected if a connected path exists between any two pixels within the region. The bibliography regarding image segmentation techniques is very extensive; therefore, we limit ourselves to a brief overview of the subject and the reader is directed to more substantial reviews, such as [2–4], for further information.

Segmentation methods can be classified as histogram based [5], graph based [6–8] or region based [9], to mention but a few general categories. The first category algorithms compute an image histogram and the regions are localized through its peaks and valleys. Thresholding techniques can obtain good segmentation, where images include two different regions because histograms disregard the spatial relationship of the image. Graph-based methods aim to extract the global image information by treating the segmentation process as a graph-partitioning problem. The region based scheme is based on the homogeneity criterion where adjacent pixels belong to the same region if they have similar features such as grey, colour or texture values. This approach is used in the proposed *I-EvoSeg* algorithm, and is described next. The split and merge method is widely used in this approach following a seeded region growing technique, where a set of initial seeds along the image are taken as input information in order to assemble the regions. The initial seeds represent the number of possible regions to be segmented. However, the problem of automatically selecting the number and positions of the initial seeds is not a trivial task because is not known a priori how many regions exist on the image and where they are located. Hence, an optimization technique could be useful to define the number and the location of regions in the segmented image.

Nowadays, evolutionary algorithms are known as a powerful stochastic optimization technique and their application to image processing and computer vision has increased mainly due to the robustness of the approach [10]. In the evolutionary computer vision community, there are a number of works dealing with image segmentation [1, 11–13]. On the other hand, Interactive Evolutionary Computation (IEC) is a general term employed in methods of evolutionary computation that use human evaluation. The idea of IEC is to involve a human expert during the on-line evaluation of the evolutionary algorithm. Human evaluation is helpful when the fitness criterion cannot be formulated explicitly, it is not well defined or sometimes when it is necessary to escape a local optimum. Thus, the interactive approach allows a faster convergence towards the most promising individuals of the population. Recent advances in IEC have initiated many attractive works, such as [11, 14–16], where the basic goal is to allow a human user to tame and the adapt random behaviour of the system into a tractable problem [17].

This chapter presents a novel approach to region-based segmentation using IEC. Our approach to image segmentation is based on analyzing texture information extracted from the Grey-Level Co-occurrence Matrix (GLCM) and combining it with the user expertise. *I-EvoSeg*, the interactive evolutionary segmentation algorithm presented in this chapter, attempts to identify how many homogeneous regions exist in the scene while it adapts the evaluation criterion by considering user interaction and texture statistics.

The remainder of this chapter is organized as follows. Section 2 describes the GLCM and texture descriptors that are used for texture analysis. Section 3 introduces the *I-EvoSeg* algorithm giving emphasis to the explanation on how interactive evolution was applied to the segmentation problem. Section 4 shows the experimental results obtained using *I-EvoSeg* and compares them with the *EvoSeg* algorithm [18]. Finally, Sect. 5 gives some concluding remarks.

2 Texture Analysis

Texture analysis is an important and useful area of study in computer vision; it comprises problems like texture classification, texture segmentation, texture synthesis and shape from texture. A given application of texture analysis usually falls into more than one category; our work is focused on texture classification and texture segmentation. The difficulty of the problem can be related to the fact that real world textures are complex to model and analyze. However, researchers agree that texture images exhibit elementary patterns that are repeated periodically within a given region. These textures can be classified as rough or smooth textures, coarse or fine textures and many others.

For instance, Hawkins [19] proposed the following texture definition: *"The notion of texture appears to depend upon three ingredients: (i) some local 'order' is repeated over a region which is large in comparison to the order's size, (ii) the order consists in the nonrandom arrangament of elementary parts, and (iii) the parts are roughly uniform entities having approximately the same dimensions everywhere within the textured region".*

Historically, the most commonly used methods for describing texture information are statistical approaches, which include first-order, second-order and higher-order statistical methods. These methods analyze the distribution of specific image properties using every pixel contained in the image. We are interested in second-order statistical methods, which represent the joint probability density of the intensity values between two pixels separated by a given vector **V**. This information is coded using the GLCM denoted by $M_{i,j}$.

Formally, the GLCM $M_{i,j}$ (π) defines a joint probability density function $f(i,j|\mathbf{V},\pi)$, where i and j are the grey levels of two pixels separated by a vector **V** , and π = **V**, R is the parameter set for $M_{i,j}$ (π). The GLCM identifies how often pixels that define a vector **V** (d, θ) and that differ by a certain amount of intensity value $\Delta = i - j$ appear in a region R of a given image $I_{L\times L}$. Here, **V** defines the distance d and orientation θ between the two pixels. The direction of **V** can or cannot be taken into account when computing the GLCM.

An example of GLCM is illustrated in Fig. 1, where the distance d is set as 1 and the direction θ is set as 0°.

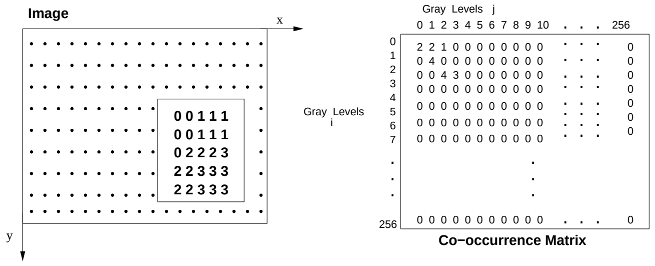
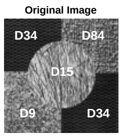


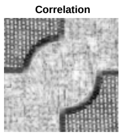
Fig. 1 GLCM example

The GLCM presents a problem when the number of different grey levels in region R increase, resulting in difficulty in handling or using directly due to the dimensions of the GLCM. Fortunately, the information encoded in the GLCM can be expressed by a varied set of statistically relevant numerical descriptors. Descriptors extracted from $M_{i,j}$ include the following: Entropy, Homogeneity, Local Homogeneity, Contrast, Moments, Inverse Moments, Uniformity, Maximum Probability, Correlation and Directivity [20, 21]. Such descriptors may be defined in the spatial domain, such as those extracted from the GLCM, or can be extracted in other frequency domains.

2.1 Texture Descriptors

Texture descriptors are computed directly from GLCM thereby reducing the dimensionality of the information that is extracted from the image I of size $L \times L$ pixels. Extracting each descriptor effectively maps the intensity values of each pixel to a new dimension. Texture descriptors used in the *I-EvoSeg* algorithm are presented next, along with an example of its corresponding image:





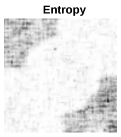
- Correlation. Correlation is a measure of grey level linear dependence between the pixels at the specified positions relative to each other. Near pixels have more correlation than far pixels.

$$S = \frac{1}{N_c.\sigma_x.\sigma_y} \mid \sum_{i}^{L-1} \sum_{j}^{L-1} (i - m_x)(j - m_y)M(i,j) \mid$$

where

$$m_x = \tfrac{1}{N_c} \textstyle\sum_i \sum_j iM(i,j)$$
$$m_y = \tfrac{1}{N_c} \textstyle\sum_i \sum_j jM(i,j)$$
$$\sigma_x^2 = \tfrac{1}{N_c} \textstyle\sum_i \sum_j (i - m_x)^2 M(i,j)$$
$$\sigma_y^2 = \tfrac{1}{N_c} \textstyle\sum_i \sum_j (j - m_y)^2 M(i,j)$$

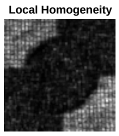
N_c is the number of occurrences in M.



- Entropy. A term more commonly found in thermodynamics or statistical mechanics. Entropy is a measure of the level of disorder in a system. Images of highly homogeneous scenes have a low associated entropy, while inhomogeneous scenes pose a high entropy measure. The GLCM entropy is obtained with the following expression:

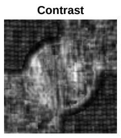
$$H = 1 - \frac{1}{N_c.ln(N_c)} \sum_{i}^{L-1} \sum_{j}^{L-1} M(i,j).ln(M(i,j)).\delta$$

where $\delta = 1$ if $M(i,j) \neq 0$ and 0 otherwise.



- Local Homogeneity. This measure provides the local similarity of the image data using a weight factor that gives small values for non-homogeneous images when $i \neq j$.

$$G = \frac{1}{N_c} \sum_{i}^{L-1} \sum_{j}^{L-1} \frac{M(i,j)}{1 + (i-j)^2}$$

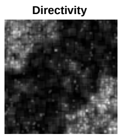


- Contrast. Opposite to homogeneity. It is a measure of the difference between intensity values of the neighbouring pixels.

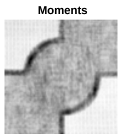
$$C = \frac{1}{N_c(L-1)^2} \sum_{k=0}^{L-1} k^2 \sum_{|i-j|=k} M(i,j)$$

- *Directivity.* This measure provides a bigger value when two pixels have the same grey level.

$$D = \frac{1}{N_c} \sum_{i}^{L-1} M(i,i)$$



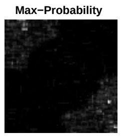
- *Moments.* It is a measure of homogeneity of an image. A homogeneous scene will contain only a few grey levels, giving a GLCM only a few but relatively high values of $M(i,j)$. This descriptor increases when the majority of the values of $M(i,j)$ are not on the diagonal. Second moment ($k = 1$) is used in this figure.



$$\mathrm{Mom_k} = \sum_{i}^{L-1} \sum_{j}^{L-1} (i-j)^k M(i,j)$$

- *Maximum Probability.* Considering the GLCM as an approximation of the joint probability density between pixels, this operator extracts the most probable difference between grey scale value pixels.

$$\max(M(i,j))$$



3 *I-EvoSeg:* Interactive Evolutionary Segmentation Algorithm

IEC is an evolutionary technique where human evaluation is applied during the optimization process. When the fitness function can be defined numerically, a canonical evolutionary algorithm is preferable. However, there are many cases where evaluating a system's performance is not trivial; in such a case, the intervention of human expertise during the selection process could be useful. This interactive approach has been explored in several research areas, such as graphic art[17], medical applications [11, 16] and image processing [22], to mention but a few. Human fatigue is a common problem in IEC that is usually solved by reducing the population size and the number of generations. In addition, a friendly graphical user interface (GUI) could also make the experience less exhaustive for the user [23].

I-EvoSeg extracts texture features from the input image in order to identify how many homogeneous regions exist while combining human evaluation and GA optimization. The idea of *I-EvoSeg* is to consider the user expertise in the GA optimization process, allowing the individual fitness to be modified on-line by the user's input. Figure 2 shows a general scheme of our algorithm which has three general stages: (i) GLCM computation (ii) calculation of texture descriptors and (iii) the segmentation process that is included in the interactive GA. *I-EvoSeg* extracts GLCM from the input image from which texture descriptors are obtained. The interactive GA randomly selects the initial seeds for each individual; subsequently, a region-based segmentation process is performed. Then, the user *interacts* with the system through a GUI (see Fig. 3). On the one hand, the user has the privilege to increase the fitness value of an individual if there exists *good* segmentation inside the current population according to his judgement. On the other hand, the user could

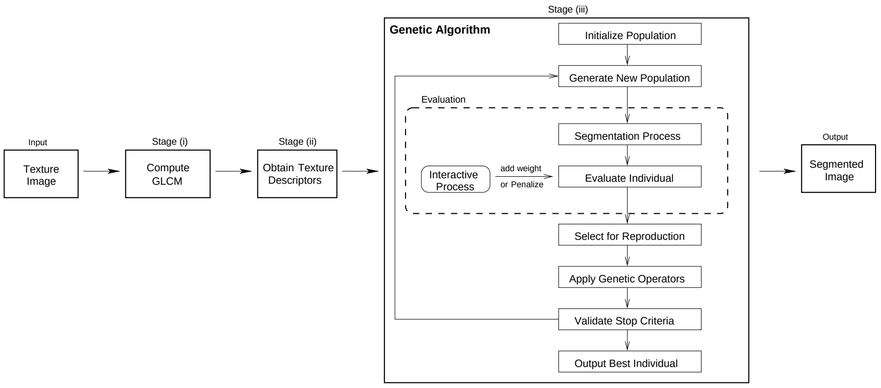
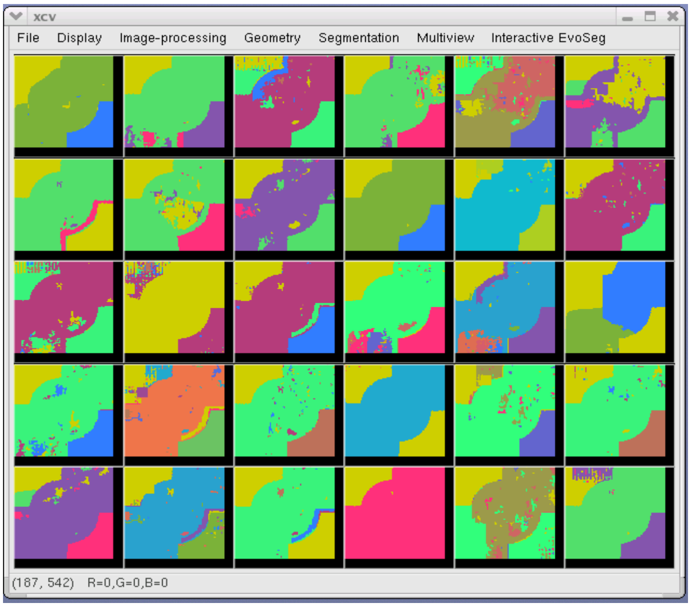


Fig. 2 *I-EvoSeg* general scheme where interaction process is included in the evaluation stage

penalize the entire population if he considers that there is not a *useful* individual in the current population. The idea of the interaction is to adapt the random behaviour of the system based on the user expertise. In this way, the algorithm is able to adapt and improve the evolutionary process.



(187, 542) R=0,G=0,B=0

Fig. 3 Graphical user interface of *I-EvoSeg*. This graphical interface shows 30 individuals as possible *good* segmentations out of 200 individuals

3.1 Co-occurrence Matrix and Texture Descriptor's Stage

GLCM's texture descriptors are used as a way of obtaining representative compact data. A large number G of intensity levels implies storing a lot of temporary data i.e., a huge $G \times G$ matrix. The GLCM is very sensitive to the size of the texture samples due to the large dimensionality of the matrix. We experimentally tested different windows sizes, directions and distances during the calculation of the GLCM using 256 grey levels. The use of small number of grey levels is equivalent to viewing the image on a coarse scale, whereas more levels reveal more image details, which requires an extra computational effort. In this way, the following parameters are used in the experiments presented in Sect. 4: the window size was set to 7×7 pixels, the direction $\theta = 0^\circ$ and the distance $d = 1$ pixel. The texture descriptors are computed from the GLCM producing a matrix for each descriptor, which is equal in size to the original image. Different texture descriptors could be used in the segmentation process as explained earlier.

3.2 Segmentation Process Embedded in an Interactive Genetic Algorithm

The segmentation process is a region-based approach where local and global similarity is taken into account in order to grow and merge regions using the texture features. The initialization of the segmentation process consists of random selection of the initial seeds, which change during the grow and merge steps; in this work, the seeds are managed as region centroids. The growing step consists of expanding each initial region using a similarity criterion based on texture information contained in the descriptor matrices. Then, the merging step is performed based on the spatial location of the regions within the image plane. The algorithm *I-EvoSeg* is explained next:

1. **Genome encoding and genetic operators.** The chromosome is represented by the position of the centroids in the image plane. Each chromosome array contains M elements that represent the possible region centroids denoted by C_i, where $i = 1, \ldots, M$. This array is updated within the segmentation process indicating the best positions of the centroids. Default parameters and operators are set, allowing an efficient exploration of the search space. The tournament selection method is used with a value of $k = 2$. The crossover is accomplished with a probability of 90%, while the mutation rate used was 10%.
2. **Segmentation process.** The segmentation process starts from the region centroids in order to classify the pixels into regions. The classification method is based on the growing and merging technique; it uses texture and spatial information in order to group pixels in possible regions. Suppose we have a set $P = \{x_1, x_2, \ldots, x_N\}$ of N pixels, where N is the total amount of pixels. Suppose we also have M disjoint subsets $R_1, R_2, \ldots, R_M$ where each subset represents a region with an associated centroid C_i. Moreover, we have a set $D_T = \{d_{T1}, d_{T2}, \ldots, d_{TN}\}$ of N descriptor values and let T be the number of texture descriptors used in the algorithm. The segmentation process is performed as follows:

a. The descriptor value s_{C_i} is calculated for each initial centroid given by the mean of the descriptor d_{pj} within a 5×5 neighbourhood around the centroid.
b. The initial class-map is formed as a template of integer numbers that represent the regions to which each pixel belongs. This class-map is created using nearest neighbour classification. Two distance measures[1] are defined in the similarity step, Δ and δ. The distance in the descriptor space is determined by Δ, whereas δ is the distance within the image plane. Thus, a pixel x_j is assigned to a region R_i if the following two conditions are true: (1) $\Delta(d_{pj}, s_{C_i}) < \Delta(d_{pj}, s_{C_q}) \ \ \forall\ q$ and (2) $\delta(x_j, C_i) < t$, where t is a threshold and $q = 1, \ldots, M$ with $q \neq i$.
c. The class-map is rearranged using a second nearest neighbour classification in order to decrease classification errors.
d. Two regions R_i and R_m are merged if they satisfy a similarity criterion based on the median of each region while considering their physical space on the image plane. Furthermore, the centroids are updated during the segmentation process when an element is added into a region. The following expression is used to update the centroids:

$$\text{centroid}(x,y) = \left(\frac{\sum_{I \varepsilon R_i} x_j}{\mid R_i \mid}, \frac{\sum_{I \varepsilon R_i} y_j}{\mid R_i \mid}\right), \tag{2}$$

where (x,y) are the pixel coordinates and $\mid R_i \mid$ is the number of elements in region R_i. In this way, the genetic information of the chromosome is adjusted to facilitate the classification of image pixels.

3. **Fitness Evaluation.** The fitness function used by *I-EvoSeg* is made of two parts: *internal fitness* and *external fitness*.

- **Internal Fitness.** It depends only on local and global minimum distances computed between regions as follows:

$$\rho = \frac{\sum_{i=1}^{c} \sqrt{\sum_{i=1}^{c} \sum_{k=1}^{n_i} (D_k - m_i)^2}}{\sum_{i=1}^{c} \sqrt{\sum_{i=1}^{c} (m_i - m)^2}}, \tag{3}$$

where c is the number of regions, D_k represents the sum of the descriptors d_{pj} of the ith region, m_i represents the median of each region, m is the total median and n_i is the number of elements in the ith region.

The internal fitness function indicates that the distances between the medians of different regions should be maximized in order to maintain the uniqueness between regions. Moreover, the distances between elements within a region should be minimized because the elements nearest to a centroid could belong to the same region.

[1] All distances measures are Euclidean.

- **External Fitness**, (***user interaction***)**.** It depends on the user decision, rewarding an individual or penalizing the current population. *I-EvoSeg* evolves a population which is displayed on a GUI for user evaluation (see Fig. 3). The two possible user decisions are as follows:
  - *Reward an individual.* The user has the option to reward an individual by adding a weight $\omega = [0,1]$ to the internal fitness when he considers that it is worthy to survive onto the next generations. The weight represents how *good* the segmentation is for the individual according to the user criterion. Therefore, if the user decides to reward an individual, the final fitness is given by

    $$\psi = \rho * (1 + \omega), \tag{4}$$

    where ρ is the internal fitness (see Eq. 3).
  - *Penalize the current population*. The user can penalize the entire population when it does not contain any worthy individuals. In such a case, $\mu = [0,1]$ is used as a penalization factor, which represents how *bad* the segmentation is within the current population according to the user expertise. Therefore, if the user decides to penalize the population, the fitness is given by

    $$\psi = \rho * (1 - \mu). \tag{5}$$

  The *I-EvoSeg* algorithm stops during the evaluation stage in order to allow interaction between the user and the system to take place. This stage helps the GA to converge towards better solutions.

4 Experimental Results

Experiments were carried out using texture images from the Brodatz database,[2] where every image has a size of 128×128 pixels. The experiments are divided into four sets: Experiment I, II, III and IV. Experimental results from *I-EvoSeg* and *EvoSeg* algorithms are compared in order to analyse the benefits of IEC. The *I-EvoSeg* algorithm uses 30 generations and 30 individuals, whereas the *EvoSeg* algorithm uses 50 generations with the same population size. Each experiment details which texture descriptors have been used and how they were selected according to the input image. Thus, we tested different combinations of texture descriptors for each texture image and chose the one that generated the best results.

4.1 Experiment I

The input image used in this experiment is depicted in Fig. 4a which has three different regions: the background and two circles; these use the D15 and D34 textures from Brodatz. The texture descriptors employed by the algorithms were *Contrast*, *Local Homogeneity*, *Directivity* and $Moment_{k=1}$. Examples of these descriptors applied to the input image are depicted in Fig. 4b–e. The *I-EvoSeg*

[2] http://sipi.usc.edu/database/

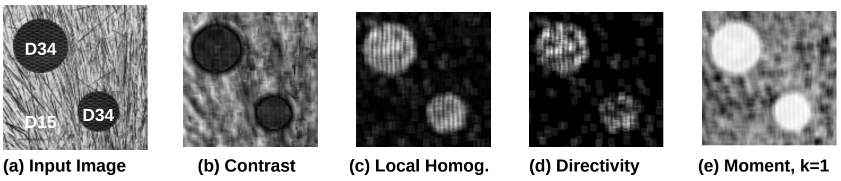


Fig. 4 **(a)** Input image using the D14 and D34 Brodatz textures. **(b–e)** Examples of the texture descriptors applied to the input image

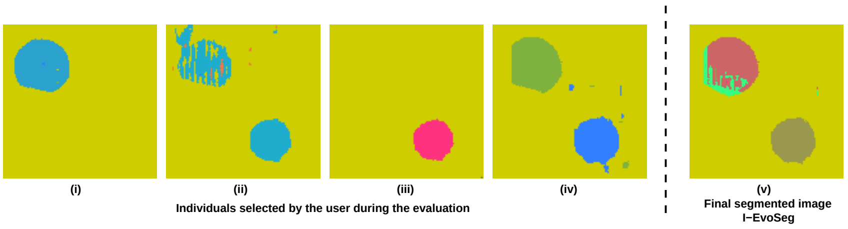


Fig. 5 **(i–iv)** Individuals selected by the user during the interactive process and **(v)** is the best segmented image using *I-EvoSeg* in Experiment I

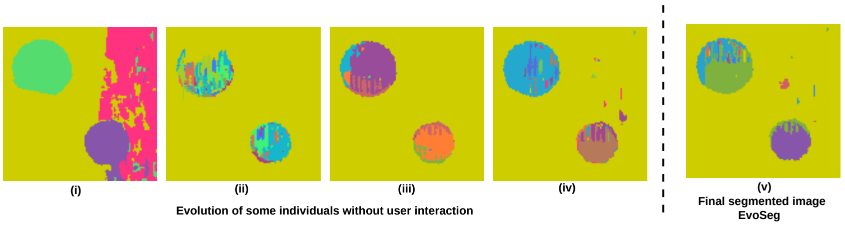


Fig. 6 **(i–iv)** show the evolution of *EvoSeg* algorithm and **(v)** represents the final segmented image

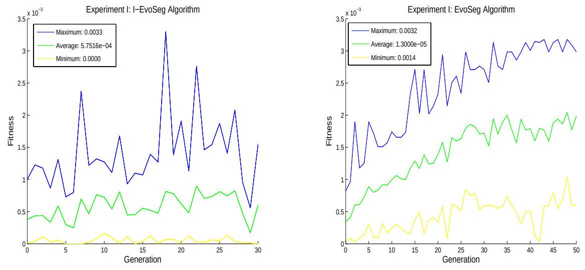


Fig. 7 *I-EvoSeg* and *EvoSeg* fitness plots corresponding to Experiment I

results are presented in Fig. 5, where some of the individuals rewarded by the user are shown, along with the final segmented image. On the other hand, Fig. 6 shows some examples of the *EvoSeg* algorithm's evolution presenting more oversegmentation in the individuals. The fitness plots for *I-EvoSeg* and *EvoSeg* are shown in Fig. 7. In these graphs, we can observe how the first *good* result occurred earlier in the evolutionary process when the user interaction is employed. The best individual was obtained with the *I-EvoSeg* algorithm in the seventh generation, whereas *EvoSeg* produced a comparable result only until generation 17. Therefore, these results suggest that interaction helps to guide the GA towards finding the best segmentation strategy in less generations.

4.2 Experiment II

The input image of the Experiment II has been used by Yoshimura and Oe [13] to illustrate works on texture image segmentation. The images depicted in Fig. 8b–h are the texture descriptors applied to the input image. Figure 9 shows how the evolution of the population is improving through the whole run. The best image segmentation using *I-EvoSeg* algorithm was obtained in the fifth generation, whereas the best segmentation using *EvoSeg* algorithm was not obtained until the 40th generation (see Fig. 11). The individual obtained in the fifth generation was used during the evolution as one of the parents. Nevertheless, no child was better than his parent in the 25 remaining generations. Examples of individuals selected by the user during

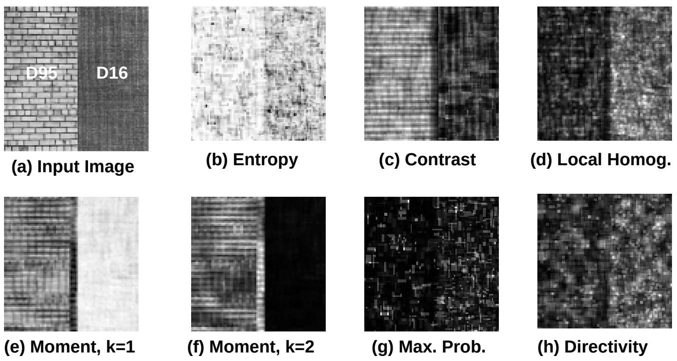


Fig. 8 **(a)** Input image for the *I-EvoSeg* and *EvoSeg* algorithms. **(b–h)** Examples of the texture descriptor images used to segment the input image

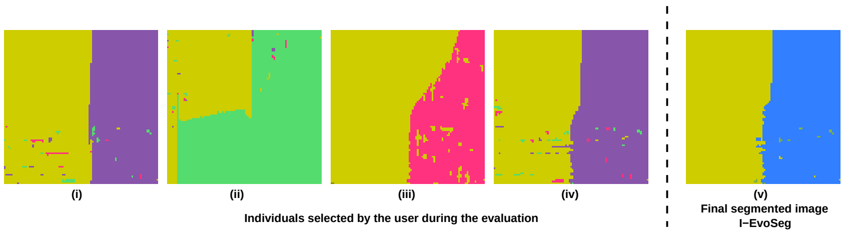


Fig. 9 **(i–iv)** Individuals selected by the user during the evolution process. **(v)** Final segmented image using *I-EvoSeg* in Experiment II

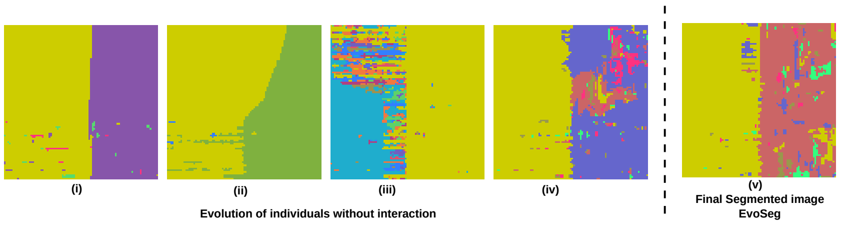


Fig. 10 **(i–iv)** Individuals obtained during the evolution using *EvoSeg*. **(v)** Illustrates the final segmented image

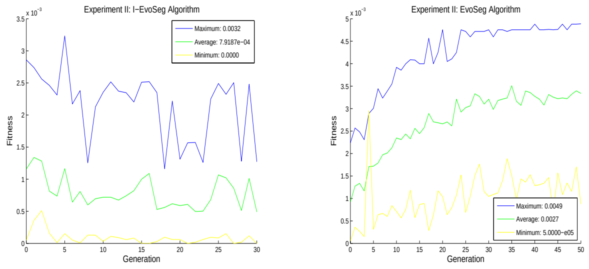


Fig. 11 Fitness plots generated by *I-EvoSeg* and *EvoSeg* algorithms that correspond to the experiment II

the interactive process are shown in Fig. 9i–iv. On the other hand, the *EvoSeg* algorithm's results depicted in Fig. 10 present an oversegmentation of the images, in other words, several misclassified regions. In this way, we could appreciate the importance of knowing which image should be selected in the interactive process in order to recognize the value of this tool.

4.3 *Experiments III and IV*

Figure 12(a) is the image used by Yoshimura and Oe [13] and it was used in our Experiments III and IV as the input image for *I-EvoSeg* and *EvoSeg* algorithms. This image is interesting because it has four different Brodatz textures (D34,D84,D15,D9) distributed into five different regions (one circle and four semi-occluded squares). It is difficult to distinguish the five regions because they contain three similar texture patterns (D9,D84,D15). Experiments III and IV using the same input image can be differentiated based on the combination of texture descriptors applied in the algorithms. First, the texture descriptors used in Experiment III were $LocalHomogeneity$, $moment_{k=1}$ and $moment_{k=2}$ (see Fig. 12b–d). Second, the texture descriptor used in Experiment IV was $moment_{k=1}$. As we observed, the results obtained by both algorithms are similar to those obtained using three descriptors or just one (see Figs. 13, 16, 14 and 17). The fitness plots are presented in

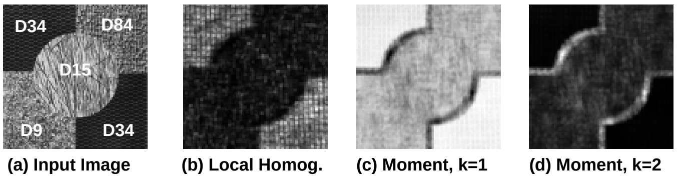


Fig. 12 **(a)** Input image used by the *I-EvoSeg* and *EvoSeg* algorithm. **(b–d)** Texture descriptors used in the experiment III. **(c)** Descriptor used in Experiment IV

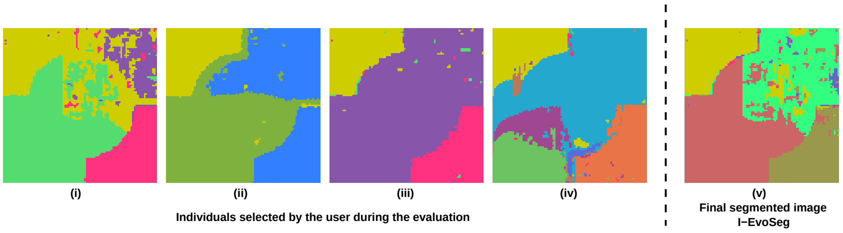


Fig. 13 **(i–iv)** Individuals selected by the user in Experiment III and **(v)** shows the segmented image

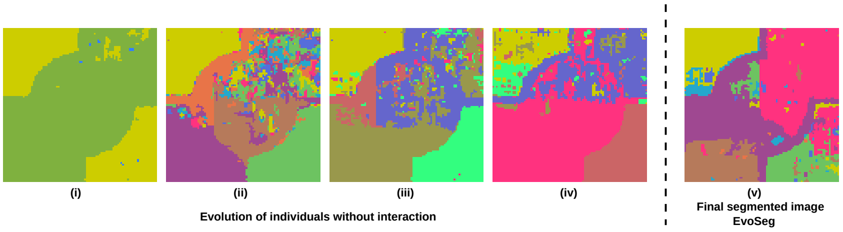


Fig. 14 **(i–iv)** Examples of individuals without the interaction process. **(v)** Segmented image obtained by the *EvoSeg* algorithm

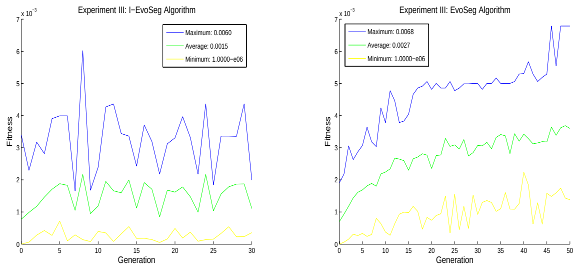


Fig. 15 Fitness plots obtained by the *I-EvoSeg* and *EvoSeg* algorithm that correspond to Experiment III

Figs. 15 and 18. We believe that it is possible to get better results by adding more generations for this particular case because we observed good promising individuals in the search space using the *I-EvoSeg* algorithm.

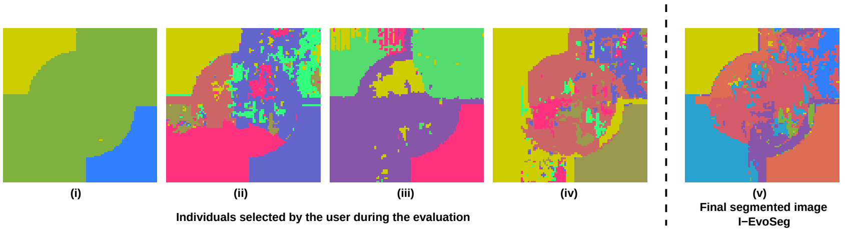


Fig. 16 **(i–iv)** Interactive individuals selected by the user during the evaluation process in Experiment IV. The image **(v)** represents the segmented image obtained by *I-EvoSeg*

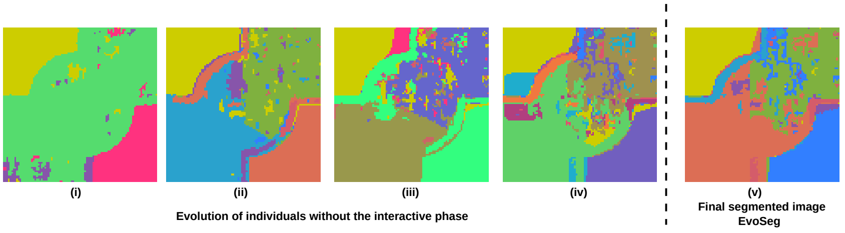


Fig. 17 **(i–iv)** Examples of individuals obtained during the evolution without the interaction process. **(v)** Segmented image obtained by the *EvoSeg* algorithm

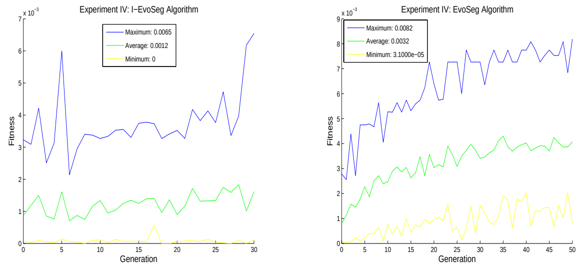


Fig. 18 *I-EvoSeg* and *EvoSeg* fitness plots corresponding to Experiment IV

5 Conclusions

This work has presented an Interactive Evolutionary Segmentation Algorithm, *I-EvoSeg*, where a significant improvement was achieved with respect to our previous work. The proposed approach uses a homogeneity criterion based on texture information, together with direct human interaction within the evolutionary loop. Interaction helps to identify the best image segmentation from multiple solutions using the user input as an external source of information in order to guide the GA towards qualitatively better solutions. We observed in the experimental results that one advantage of using the interactive process is that improvements on texture segmentation are achieved in earliest generations. The approach was based on GLCM information using statistical descriptors to propose an internal fitness where local and global criteria have been considered and an external fitness that depends on the user expertise. We tested *I-EvoSeg* using well-known texture images and compared with the EvoSeg algorithm. Experimental results show that the interactive approach produces qualitatively better segmentation with a lower computational effort. In future research, it would be interesting to analyze which texture descriptors could represent the best combination of texture patterns to produce a better image segmentation.

Acknowledgements. This research was funded by Ministerio de Ciencia y Tecnología, and by CONACyT and INRIA through the LAFMI project. First author was supported by scholarship 0416442 from CONACyT.

References

1. Bhandarkar, S., Zhang, H.: Image segmentation using evolutionary computation. IEEE Transactions on Evolutionary Computation 3(1), 1–21 (1999)
2. Freixenet, J., Muñoz, X., Raba, D., Martí, J., Cufí, X.: Yet another survey on image segmentation: region and boundary information integration. In: Heyden, A., Sparr, G., Nielsen, M., Johansen, P. (eds.) ECCV 2002. LNCS, vol. 2352, pp. 408–422. Springer, Heidelberg (2002)
3. Haralick, R., Shapiro, L.: Survey: Image segmentation techniques. Computer Vision, Graphics, and Image Processing 29(1), 100–132 (1985)
4. Pal, N.R., Pal, S.K.: A review on image segmentation techniques. Pattern Recognition 26(9), 1277–1294 (1993)
5. Lim, Y.W., Lee, S.U.: On the color image segmentation algorithm based on the thresholding and the fuzzy C-means technique. Pattern Recognition 23(9), 935–952 (1990)
6. Pavan, M., Pelillo, M.: A new graph-theoretic approach to clustering and segmentation. In: IEEE Computer Society Conference on Computer Vision and Pattern Recognition, vol. 1, pp. 145–152 (2003)
7. Duarte, A., Sánchez, A., Fernández, F., Montemayor, A.: Improving image segmentation quality through effective region merging using hierarchical social metaheuristic. Evolutionary Computer Vision and Image Understanding 27(11), 1239–1251 (2006)
8. Shi, J., Malik, J.: Normalized cuts and image segmentation. IEEE Transactions on Pattern Analysis and Machine Intelligence 32(8), 888–905 (2000)
9. Deng, Y., Manjunath, B.: Unsupervised segmentation of color-texture regions in images and video. IEEE Transactions on Pattern Analysis and Machine Intelligence 23(8), 800–810 (2001)
10. Olague, G., Cagnoni, S., Lutton, E.: Introduction to the special issue on evolutionary computer vision and image understanding. Pattern Recognition Letters 27(11), 1161–1163 (2006)
11. Cagnoni, S., Dobrzeniecki, A., Poli, R., Yanch, J.: Genetic algorithm-based interactive segmentation of 3D medical images. Image and Vision Computing 17(12), 881–895 (1999)
12. Bhanu, B., Lee, S., Ming, J.: Adaptive image segmentation using a genetic algorithm. IEEE Transactions on Systems, Man, and Cybernetics 25(12), 1543–1567 (1995)

13. Yoshimura, M., Oe, S.: Evolutionary segmentation of texture image using genetic algorithms towards automatic decision of optimum number of segmentation areas. Pattern Recognition 32, 2041–2054 (1999)
14. Hee-Su, K., Sung-Bae, C.: Application of interactive genetic algorithm to fashion design. Engineering Applications of Artificial Intelligence 13, 635–644 (2000)
15. Lutton, E., Grenier, P., Levy-Vehel, J.: An interactive EA for multifractal bayesian denoising. In: Rothlauf, F., Branke, J., Cagnoni, S., Corne, D.W., Drechsler, R., Jin, Y., Machado, P., Marchiori, E., Romero, J., Smith, G.D., Squillero, G. (eds.) EvoWorkshops 2005. LNCS, vol. 3449, pp. 274–283. Springer, Heidelberg (2005)
16. Legrand, P., Bourgeois-Republique, C., Pean, V., Harboun-Cohen, E., Levy-Vehel, J., Frachet, B., Lutton, E., Collet, P.: Interactive evolution for cochlear implants fitting. Genetic Programming and Evolvable Machines 8(4), 319–354 (2006)
17. Lutton, E.: Evolution of fractal shapes for artists and designers. International Journal on Artificial Intelligence Tools 15(4), 651–672 (2006)
18. Perez, C., Olague, G.: Unsupervised evolutionary segmentation algorithm based on texture analysis. In: Giacobini, M. (ed.) EvoWorkshops 2007. LNCS, vol. 4448, pp. 407–414. Springer, Heidelberg (2007)
19. Hawkins, J.: Textural properties for pattern recognition. In: Lipkin, B., Rosenfeld, A. (eds.) Picture Processing and Psychopictorics. Academic Press, New York (1969)
20. Parker, J.: Algorithms for image processing and computer vision. John Wiley, New York (1996)
21. Haralick, R., Shanmugam, K., Dinstein, I.: Textural features for image classification. IEEE Transactions on System, Man and Cibernetics 3(6), 610–621 (1973)
22. Takagi, H.: Interactive evolutionary computation: Fusion of the capabilities of EC optimization and human evaluation. Proceedings of the IEEE 89(9), 1275–1296 (2001)
23. Quiroz, J., Louis, S., Dascalu, S.: Interactive evolution of XUL user interfaces. In: 9th Conference on Genetic and Evolutionary Computation, GECCO 2007, pp. 2151–2158 (2007)

Detecting Scale-Invariant Regions Using Evolved Image Operators

Leonardo Trujillo and Gustavo Olague

Abstract. This chapter describes scale-invariant region detectors that are based on image operators synthesized through Genetic Programming (GP). Interesting or salient regions on an image are of considerable usefulness within a broad range of vision problems, including, but not limited to, stereo vision, object detection and recognition, image registration and content-based image retrieval. A GP-based framework is described where candidate image operators are synthesized by employing a fitness measure that promotes the detection of stable and dispersed image features, both of which are highly desirable properties. After a significant number of experimental runs, a plateau of maxima was identified within the search space that contained operators that are similar, in structure and/or functionality, to basic LoG or DoG filters. Two such operators with the simplest structure were selected and embedded within a linear scale space, thereby making scale-invariant feature detection a straightforward task. The proposed scale-invariant detectors exhibit a high performance on standard tests when compared with state-of-the-art techniques. The experimental results exhibit the ability of GP to construct highly reusable code for a well known and hard task when an appropriate optimization problem is framed.

1 Introduction

In the early years of computer vision research, recognition systems applied three sequential stages that were thought to be essential: feature extraction, perceptual grouping and model correspondence [1]. With time, however, researchers have recognized that perceptual grouping, or segmentation, represents an ill-posed mathematical problem [2]. Thence, since the late 1990s, many of the recognition

Leonardo Trujillo and Gustavo Olague
Centro de Investigación Científica y de Educación Superior de Ensenada, Km. 107 Carretera Tijuana-Ensenada, 22860, Ensenada, BC, México
e-mail: `leonardo.trujillo.ttl@gmail.com,olague@cicese.mx`
`http://ciencomp.cicese.mx/evovision`

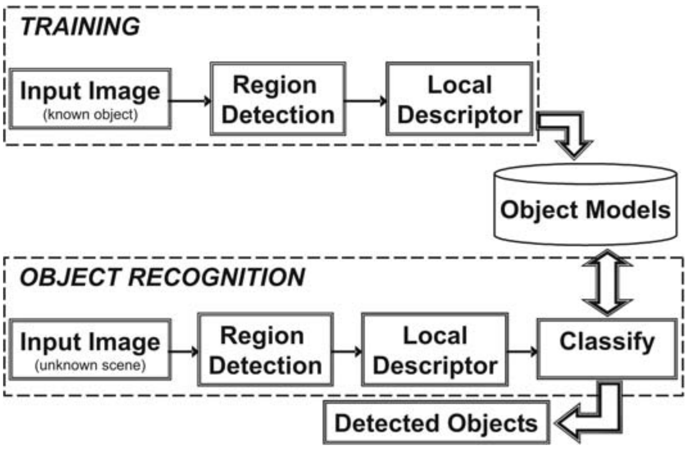


Fig. 1 Simplified model for vision systems that are based on local features

systems have adopted a simplified model for low-level vision processes [3]. Two such approaches have received emphasis, one is a holistic approach to image recognition [4–6] and the other employs a sparse representation based on locally prominent regions [7–10]. This work is concerned with image operations related to the latter (see Fig. 1).

The sparse approach, from a high-level view, includes two stages that are common to many machine-learning systems, one for training and another for testing. During training, regions are extracted from an image using stable and robust operators [3, 11–14] (see Fig. 2). These type of operators represent the main topic of discussion of the present chapter. Then, a representative descriptor is built for each of the extracted regions; such descriptors are expected to be distinctive and invariant [7, 14, 15]. Finally, the set of detected regions and corresponding descriptors are used to build a model for the imaged scene. During testing, the same process is repeated; however, the new set of extracted features is, in this case, compared with stored models, and when appropriate matching criteria are met, it is possible to recognize known scenes or objects.

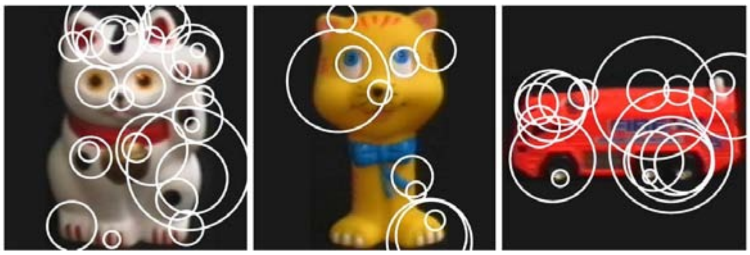


Fig. 2 Detected interest regions taken from three images in the COIL-100 data set

Insofar as this approach relies on invariant region detection and description, it is considered to be robust to partial occlusions while avoiding the need for explicit image segmentation. Therefore, it is paramount to detect stable image features, also known as interest regions. In this chapter, these regions can refer to either single image pixels or larger image patches depending on the type of detection employed. The simplest description for interest regions is to regard them as image patches that exhibit a high level of variation or irregularity with respect to a particular local measure computed by a specific operator. Based on this informal definition, different region detectors define different operators that compute an interest measure for each image pixel [3, 12, 13]. Then, local extrema of this response, in both scale and location, are selected as interest regions. It is important to note that a *detector* refers to the complete algorithmic process that identifies interest regions; on the other hand, an *operator* is a function that extracts an interest measure.

Therefore, a system designer is left with an important decision: which detector is the most appropriate for a particular vision system? Answering this question is not trivial and careful consideration must be given before choosing appropriately. However, a widely accepted performance measure exists, which allows objective comparisons to be made between different detectors. The performance measure is called the *repeatability rate* and it quantifies the stability of a detector under changes to imaging conditions [3]. These changes are modelled as image transformations, which include *translation, illumination change, image rotation, scale change and projective transformations*. Detectors invariant to the first three types of image transformations are interest point detectors [11–13, 16], invariance to the first four is achieved by scale-invariant region detectors [17, 18] and affine covariant feature detectors are invariant to all [19].

Previous work by Trujillo and Olague [12, 13] proposed a novel approach to construct optimized interest point detectors using Genetic Programming (GP) [20, 21] as an optimization engine and the repeatability rate as part of the fitness function. That work provided a significant improvement over previous attempts to synthesize a competitive detector through GP [22, 23]. This chapter describes an extension of that research [18], where evolved operators are embedded into a linear scale space, thereby facilitating the detection of scale-invariant regions [24].

Several topics related to feature detection are explored throughout this chapter. A conceptual search space is described where different types of image operators reside, some of them useful for the described task. Furthermore, an interesting result of the evolutionary process is identified: the fact that it tends to converge to regions in search space where simple and well-known image operations lie. Finally, this chapter describes two scale-invariant region detectors based on GP-evolved operators [18], the algorithmic implementation of which is simpler than previous designs and achieves no worse than equal, and sometimes better, performance than the state-of-the-art in the field.

The remainder of the chapter is organized as follows. Section 2 introduces relevant theoretical background regarding interest region detection, performance criteria and a process by which GP is able to evolve image operators. Section 3 discusses what scale-invariant detection is and how it can be performend, reviews relevant state-of-the-art and introduces the proposed detection algorithms. In Sect. 4, the experimental set up is described along with results of benchmark tests. Finally, concluding remarks are given in Sect. 5 along with an outline of possible future work.

2 Background

This section describes common interest operators and how GP can be used to automatically synthesize operators that are capable of detecting interesting image pixels.

2.1 Interest Operators

The following discussion centres on operators that were developed for interest point detection, the simplest type of interest region (see Fig. 3). However, these interest measures can also be used to construct scale-invariant region detectors using an appropriate problem modelling [17]. What is required is an analytical perspective that accounts for scale-related transformations, and this framework is provided by the concept of scale space; more on this topic is discussed in Sect. 3. Nevertheless, the concepts described and the desired traits presented for interest point detection can be extrapolated quite easily to the more general scale-invariant regions without any loss of generality [17].

Interest point detection resulted from research devoted to corner detection. A common taxonomy contains three principal classes: *contour-based methods* [25, 26], *parametric model-based methods* [27, 28] and *intensity-based methods* [11, 16, 29–33]. The class of corner detectors that operate directly on the intensity image are more appropriately referred to as *interest point detectors*. A measure of how interesting an image region is can be computed using a mapping $K : \Re^+ \rightarrow \Re$, or interest operator, where different region detectors employ a different K. Applying K to an image I yields what can be understood as an *interest image* I^*. Then, most detectors follow the same process: (1) non-maxima suppression that eliminates pixels that are not local maxima and (2) a thresholding step that obtains the final set of points; both of the final steps are tuned empirically and can be seen in Fig. 3.

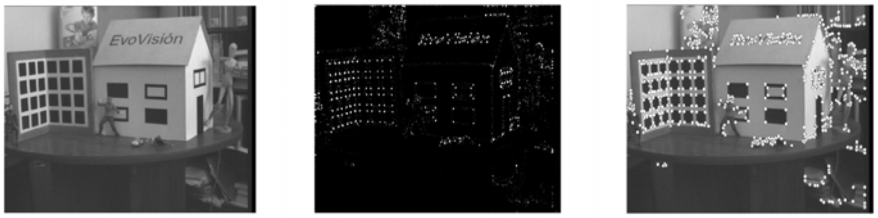


Fig. 3 A look at interest point detection: **Left,** an input image I; **Middle,** *interest image* I^*; **Right,** detected points after non-maximum suppression and thresholding superimposed on I

To begin with, some operators base their interest measure K on the local autocorrelation matrix A, which characterizes the gradient distribution around each pixel with

$$A(\mathbf{x},\sigma_{\mathrm{I}},\sigma_{\mathrm{D}}) = \sigma_{\mathrm{D}}^2 \cdot G_{\sigma_{\mathrm{I}}} * \begin{bmatrix} L_x^2(\mathbf{x},\sigma_{\mathrm{D}}) & L_x(\mathbf{x},\sigma_{\mathrm{D}})L_y(\mathbf{x},\sigma_{\mathrm{D}}) \\ L_x(\mathbf{x},\sigma_{\mathrm{D}})L_y(\mathbf{x},\sigma_{\mathrm{D}}) & L_y^2(\mathbf{x},\sigma_{\mathrm{D}}) \end{bmatrix} , \quad (1)$$

where σ_{D} and σ_{I} are the derivation and integration scales, respectively; $L_u(\mathbf{x},\sigma_{\mathrm{D}})$ is the Gaussian derivative in direction u of image I at point $\mathbf{x}$ given by

$$L_u(\mathbf{x},\sigma_{\mathrm{D}}) = \frac{\delta}{\delta u} G_{\sigma_{\mathrm{D}}} * I(\mathbf{x}) ,$$

G_σ is a Gaussian smoothing function with standard deviation σ ($\sigma_{\mathrm{D}} = 1$ is used unless noted otherwise). Detectors based on A include those proposed by Harris and Stephens [11], Förstner [32] and Shi and Tomasi [33], with

$$K_{\mathrm{Harris\&Stephens}}(\mathbf{x}) = det(A) - k \cdot Tr(A)^2 ,$$

$$K_{\mathrm{Forstner}}(\mathbf{x}) = \frac{det(A)}{Tr(A)} ,$$

$$K_{\mathrm{Shi\&Tomasi}}(\mathbf{x}) = \min\{\lambda_1, \lambda_2\} ,$$

where $\lambda_1$and λ_2 are the two eigenvalues of A. The definition of A is as in [3], used with the so-called *Improved Harris* detector. Beaudet [16] proposed the determinant of the Hessian, which is proportional to the Gaussian curvature, as an interest measure

$$K_{\mathrm{Beaudet}}(\mathbf{x}) = I_{xx}(\mathbf{x}) \cdot I_{yy}(\mathbf{x}) - I_{xy}^2(\mathbf{x}) ,$$

where $I_u(\mathbf{x})$ is the image derivative in direction u. Wang and Brady [31] characterized the curvature response using the Laplacian and the gradient magnitude,

$$K_{\mathrm{Wang\&Brady}}(\mathbf{x}) = (\nabla^2 I(\mathbf{x}))^2 - s|\nabla I(\mathbf{x})|^2 .$$

Kitchen and Rosenfeld [30] presented an interest measure aimed at detecting image corners, which is given by the product between the gradient magnitude and the magnitude of the change of direction of the gradient,

$$K_{\mathrm{Kitchen\&Rosenfeld}}(\mathbf{x}) = \frac{I_{xx}(\mathbf{x})I_y^2(\mathbf{x}) + I_{yy}(\mathbf{x})I_x^2(\mathbf{x}) - 2I_{xy}(\mathbf{x})I_y(\mathbf{x})I_x(\mathbf{x})}{I_x^2(\mathbf{x}) + I_y^2(\mathbf{x})} .$$

Smith and Brady [34] designed the SUSAN detector, which deviates from the detectors described up to now, in that it does not base its interest measure on a derivative-based image operator and instead employs a geometric criteria that explicitly searches for image corners. However, it has been reported to perform poorly in some comparative tests [35].

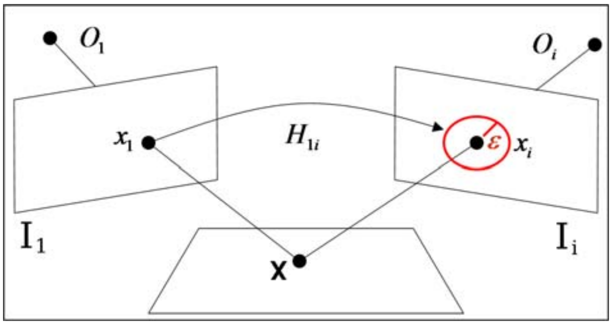


Fig. 4 A 3D point is projected onto points x_1 and x_2 on images I_1 and I_2, respectively. Point x_1 is said to be repeated by x_i if a point is detected within a neighbourhood of x_i of size ε. For planar scenes, x_1 and x_i are related by the homography $H_{1,i}$

2.2 Performance Criteria

The main properties that interest points, and more generally interest regions, are expected to fulfil are that they must be invariant to image transformations and must represent a sparse sampling of the image, thus suggesting that they are *unique* and *stable* within an image. Therefore, before describing the GP approach used to synthesize interest operators, two measures are presented that quantify these properties and are, subsequently, used as the primary optimization criteria: the *repeatability rate* and the amount of *point dispersion*.

2.2.1 Repeatability Rate

The performance measure employed to evaluate an operator's stability is the repeatability rate that captures how detection is independent of the imaging conditions [3]. An interest point x_1, detected in image I_1, is repeated in image I_i if the corresponding point x_i is detected in image I_i. In the case of planar scenes, a relation between points x_1 and x_i can be established with the homography $H_{1,i}$, where

$$x_i = H_{1,i}x_1 \ . \tag{2}$$

The repeatability rate measures the number of repeated points between both images with respect to the total number of detected points. However, parts of image I_1 may not appear on the transformed image I_i. In order to account for this, repeated and detected points are only counted if they lie in the common parts of image I_1 and image I_i. Furthermore, a small amount of detection error needs to be taken into account, because exact localization is not paramount for all applications. A repeated point is said to be detected at pixel x_i if it lies within a given neighbourhood of x_i of size ε; all the experiments use $\varepsilon = 1.5$ pixels (see Fig. 4). The set of point pairs (x_1^c, x_i^c) that lie in the common part of both images and correspond within an error ε is defined by

$$R_{I_i}(\varepsilon) = \{(x_1^c, x_i^c)\,|dist\,(H_{1,i}x_1^c, x_i^c) < \varepsilon\} \ . \tag{3}$$

Thus, the repeatability rate $r_i(\varepsilon)$ of points extracted from image I_i with respect to points from image I_1 is defined by the following equation:

$$r_{I_i}(\varepsilon) = \frac{|R_{I_i}(\varepsilon)|}{min(\gamma_1, \gamma_i)} , \tag{4}$$

where $\gamma_1 = |\{x_1^c\}|$ and $\gamma_i = |\{x_i^c\}|$ are the total number of points extracted from image I_1 and image I_i, respectively. The ratio r_{I_i} employs the minimum of detected points because it is assumed that when the transformation $H_{1,i}$ causes $\gamma_1 \neq \gamma_i$, the minimum represents an upper bound on the total number of repeatable points. Results presented in [3] suggest that the *Improved-Harris* operator has the best stability, outperforming every other detector included in the survey. These results have contributed into making the Harris detector the most widely used in computer vision applications. However, the list of compared detectors is not exhaustive; for example, it leaves out [16] and [30].

2.2.2 Point Dispersion

Global separability between extracted points suggests that *most images* will have interest points that are not crowded together on isolated regions of the image plane. Furthermore, if interest points are spread out across the image, then it is more likely that said points will be determinately different amongst themselves and when compared with the rest of the image pixels. Obviously, this property is image dependent and requires a priori knowledge of the expected number of points and their distribution within the image plane. Nevertheless, it is a useful measure for how well an interest operator is able to sample the information contained within the image. When considering the property of global separability, a metric that quantifies the amount of point dispersion can be obtained using an entropy measure. Hence, entropy is computed from the partition $\{\mathscr{I}_j\}$ of the spatial dimensions of the image plane I. Therefore, $\mathscr{D}$ is the entropy value of the spatial distribution of detected interest points within the image plane, given by

$$\mathscr{D}(I,X) = -\sum P_j \cdot \log_2(P_j) , \tag{5}$$

where P_j is approximated by the 2D histogram of the position of points found in I. The image is divided into a 2D grid, where each bin has a size of 8×8 pixels. The probabilities are computed by $P_j = \frac{\Gamma(x,y)}{\gamma}$, where $\Gamma(x,y)$ is the number of interest points contained within bin (x,y) and γ is the total number of detected points.

2.3 Constructing Interest Operators with GP

Trujillo and Olague [12, 13, 36] presented a GP-based approach for the automatic synthesis of interest operators used for interest point detection. Defining an operator K was formulated as an optimization problem where an aggregate objective function $f(K)$ promotes a high average repeatability rate $r_{K,J}(\varepsilon)$ on a training image sequence J. The training sequence J includes a reference image I_0 and m test images $J = [I_0, I_1, ..., I_m]$. Furthermore, $f(K)$ also promotes a high point dispersion $\mathscr{D}$ on the reference image I_0. The Function and Terminal sets, F and T respectively, include primitive operations common in many interest operators (see Sect. 2.1) such that

$$\begin{aligned} F = &\left\{+, |+|, -, |-|, |I_{\text{out}}|, *, \div, I_{\text{out}}^2, \sqrt{I_{\text{out}}}, \log_2(I_{\text{out}}), EQ(I_{\text{out}}), k \cdot I_{\text{out}}\right\} \\ &\bigcup \left\{\frac{\delta}{\delta x} G_{\sigma_{\text{D}}}, \frac{\delta}{\delta y} G_{\sigma_{\text{D}}}, G_{\sigma=1}, G_{\sigma=2}\right\} , \\ T = &\left\{I, L_x, L_{xx}, L_{xy}, L_{yy}, L_y\right\} , \end{aligned} \tag{6}$$

where I is the input image, and I_{out} can be any of the terminals in T as well as the output of any of the functions in F; $EQ(I)$ is an histogram normalization operation; L_u are Gaussian image derivatives along direction u; G_σ are Gaussian smoothing filters; $\frac{\delta}{\delta u} G_{\sigma_{\text{D}}}$ represents the derivative of a Gaussian function and the constant $k = 0.05$. The aggregate objective function was defined as

$$f(K) = r_{K,J}(\varepsilon) \cdot \phi_x^\alpha \cdot \phi_y^\beta \cdot N_\%^\gamma , \tag{7}$$

where the exponential weights are set to: $\alpha = 20$, $\beta = 20$ and $\gamma = 2$. The terms ϕ_u promote *global separability*; they behave like sigmoidal functions in a specified interval,

$$\phi_u = \begin{cases} \dfrac{1}{1 + e^{-a_u(\mathscr{D}_u - c_u)}}, & \text{when} \quad \mathscr{D}_u < \mathscr{D}_u^{\max} , \\ 0 & \text{otherwise} . \end{cases} \tag{8}$$

$\mathscr{D}_u$ is the dispersion along direction u on the reference image I_0. Values for ϕ_u are thresholded in order to discourage individuals that exploit the separability term, with experimentally chosen values set for $D_y^{\max} = 5$ and $D_x^{\max} = 5.9$, and $a_x = 7$, $c_x = 5.05$, $a_y = 6$ and $c_y = 4.3$. The final term,

$$N_\% = \left(\frac{\text{extracted points}}{\text{requested points}}\right)^\gamma , \tag{9}$$

is a penalizing factor that reduces the fitness value for detectors that return less than the number of requested points, set to 500 for the training set employed.

2.4 Conceptual Search Space

Figure 5 presents a high-level view of the space Ω of possible interest operators constructed with primitives from $\{F \cup T\}$. This conceptual space can be a useful tool to comprehend what the GP process considers during its search by further partitioning Ω into a reasonable set of easily identifiable subspaces. One possible subspace is $\Omega_\delta \subset \Omega$, which contains operators that use image derivatives, taken from T, to obtain an interest measure. Figure 5 also shows the subspace of operators that rely on measures based on the local autocorrelation matrix Ω_A; due to the manner in which A has been defined, the relation $\Omega_A \subset \Omega_\delta$ is true. Both Ω_A and Ω_δ group

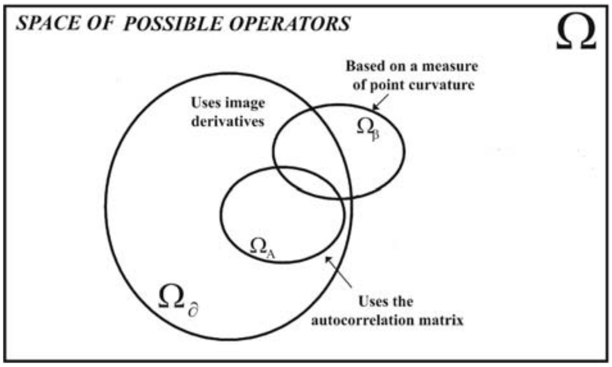

SPACE OF POSSIBLE OPERATORS
Ω
Based on a measure of point curvature
Uses image derivatives
Ω_β
Ω_A
Ω_∂
Uses the autocorrelation matrix

Fig. 5 Space of possible interest operators constructed with $\{F \cup T\}$

operators are based on their structure and not their functionality, in other words, based on their genotype and not on their phenotype. On the other hand, a subspace $\Omega_\beta \subset \Omega$ contains operators that extract a measure related to surface curvature. In this case, Ω_β contains operators with similar functionality, and its intersection with other subspaces, those based on structure, may or may not be empty. When a one-to-one correspondence between structure and functionality cannot be expected [37], the distinction is essential. For example, consider function approximations, such as the response of a Laplacian-of-Gaussian (LoG) filter approximated by a Difference-of-Gaussian (DoG) filter, two operators that are quite different based on their genotype while at the same time have similar functionality.

2.5 Previous Results

Trujillo and Olague [12, 13] presented two operators synthesized with the GP search described above: K_{IPGP1} and K_{IPGP2}; the acronym *IPGP* represents *Interest Point detection with Genetic Programming*. These operators outperformed or matched all previous manmade designs on standard tests [38], and are defined as

$$K_{IPGP1} = G_{\sigma=2} * (G_{\sigma=1} * I - I), \tag{10}$$

$$K_{IPGP2} = G_{\sigma=1} * (L_{xx}(\mathbf{x}, \sigma_D = 1) \cdot L_{yy}(\mathbf{x}, \sigma_D = 1) - L_{xy}^2(\mathbf{x}, \sigma_D = 1)). \tag{11}$$

K_{IPGP1} identifies corners, edges and blobs with salient low-intensity regions. Its additive inverse extracts salient high-intensity regions. K_{IPGP2}, on the other hand, is a modified version of the operator proposed by Beaudet [16]. Figure 6 presents sample points extracted with K_{IPGP2}, and shows how they can be used in the stereo correspondence problem.

Further experimental runs of the GP suggest that a plateau of local maxima exists within the neighbourhood of K_{IPGP1} because several runs produced operators that are genetically similar to K_{IPGP1}. A total of 30 additional runs were carried out, from which 12 failed to converge to useful results, and the remaining 18 are shown in Ta-

Table 1 Evolved interest operators. The Table is divided into three subsets that group operators in accordance with their genetic material. In addition, the Table is made up of five columns: Name (the identification name for each operator); Operator (the mathematical expression); Run (the run numbers in which the operator was produced); Fitness (the fitness of the operator) and $r_J\star$ (the average repeatability rate computed on the training sequence)

Name	**Operator**	**Run**	**Fitness**	$r_J\star$
$IPGP1^*$	$G_2 * \lvert I - G_2 * I\rvert^2$	3, 28	77.91	94.78
$IPGP3$	$G_1 * G_1 * G_1 * G_2 * G_2 * (G_1 * I - I)$	5	83.98	95.9
$IPGP4$	$G_2 * G_2 * G_2 * (G_2 * I - I)$	7	85.98	96.35
$IPGP5$	$G_1 * G_2 * \lvert I - G_1 * I\rvert^2$	19	83.86	93.22
$IPGP6$	$G_2 * G_2 * G_1 * \left(\frac{I}{G_2 * I}\right)$	15	78.13	94.84
$IPGP7$	$G_2 * \lvert 2 \cdot L_{yy} + 2 \cdot L_{xx}\rvert$	4, 9	78.33	94.92
$IPGP8$	$G_2 * \left\lvert L_{xx} + 2 \cdot G_2 (L_{xx} + L_{yy})^2\right\rvert$	11	73.86	90.54
$IPGP9$	$G_2 * G_2 * \lvert 2 \cdot L_{yy} + 2 \cdot L_{xx} + L_{xy}\rvert$	16	80.3	93.44
$IPGP10$	$G_2 * \lvert L_{yy} + L_{xx}\rvert$	18, 20	77	92.81
$IPGP11$	$G_1 * \left(\frac{G_1 * I}{(G_1 * G_1 * I)^3}\right)$	21	78.23	92.44
$IPGP12$	$\frac{G_2 * I^{\frac{3}{2}}}{(G_1 * I)^{\frac{9}{4}}}$	22	72.67	91.91
$IPGP13$	$G_2 * G_2 * [(G_2 * I)(G_2 * G_1 * I - I)]$	6	85.72	96.37
$IPGP14$	$G_2 * G_2 * [(G_2 * G_2 * G_2 * I)(G_2 * G_2 * I - I)]$	23	85.94	96.5
$IPGP15$	$\frac{G_2 * [G_2 * G_2 * \lvert I - G_1 * I - G_2 * G_2 * I\rvert]}{G_2 * G_2 * I}$	24	85.81	95.15
$IPGP16$	$\frac{G_2 * G_2 * [G_1 * G_2 * I^2 - I^2]}{G_2 * G_2 * G_1 * I}$	30	84.63	96.8
	$\star$ To obtain r_J, the size of the neighbourhood **W** for non-maximum suppression was set to $n = 2$.			

ble 1 (see [36]). Only 15 distinct operators were obtained, however, because some of the operators were discovered in more than one run. The operators in Table 1 are grouped together into three subgroups based on genetic similarity between them: (1)

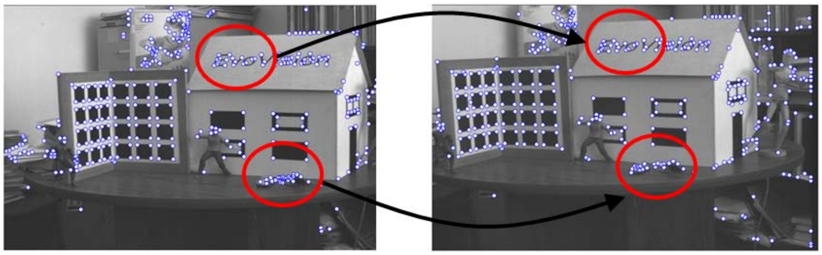


Fig. 6 A stereo pair taken at the EvoVisión Laboratory; notice how interest points can be used to compute the correspondence between both images, which are repeatable. Interest points are extracted with the K_{IPGP2} operator [12, 13]

the first group contains operators similar to K_{IPGP1}; (2) the next group contains operators that employ image derivatives and specifically perform a Laplacian operation and the final group (3) contains a variety of unorthodox and structurally large operators. The first group is of interest here due to the simplicity and high performance scores. Furthermore, groups 1 and 2 can be taken to be functionally equivalent, emphasizing more the type of image operations preferred by the GP search process. From these, K_{IPGP1^*} is chosen because it represents a close neighbour, both in the function space and fitness space, of K_{IPGP1}.

$$K_{IPGP1^*} = G_{\sigma=2} * |G_{\sigma=1} * I - I|^2. \tag{12}$$

K_{IPGP1^*} identifies maxima related to both K_{IPGP1} and its additive inverse.

Proposition 1. *Both K_{IPGP1} and K_{IPGP1^*} are proportional to DoG (Difference-off-Gaussian) filters, if we assume that image I is derived from an unknown image $\hat{I}$ blurred with a Gaussian of unknown standard deviation $\hat{\sigma}$ such that $I = G_{\hat{\sigma}} * \hat{I}$ and*

$$G_\sigma * I - I = G_\sigma * G_{\hat{\sigma}} * \hat{I} - G_{\hat{\sigma}} * \hat{I} \propto G_{\sigma+\hat{\sigma}} * \hat{I} - G_{\hat{\sigma}} * \hat{I} = (G_{\sigma+\hat{\sigma}} - G_{\hat{\sigma}}) * \hat{I}. \tag{13}$$

Based on Proposition 1, K_{IPGP1} and K_{IPGP1^*} are approximations of the 2D LoG function or of the Mexican hat wavelet (see Fig. 7). In image processing, these type of operators are often used for edge detection. However, some authors have employed similar operators to detect interesting image regions [14, 24]. Furthermore, the methodology described in this chapter is meant to promote the discovery of operators that fulfil a certain set of performance criteria without any restrictions on how these operators have been used before. Thus, some of the operators presented in Table 1, for instance, may seem unorthodox or counter intuitive.

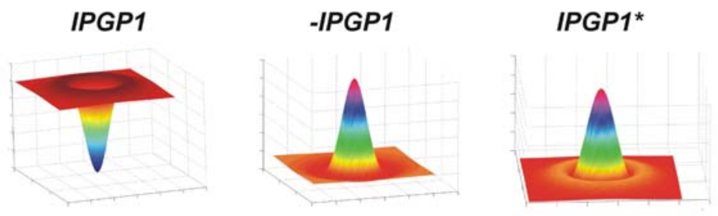


Fig. 7 Proposed scale-invariant interest operators

3 Scale-Invariant Region Detection

The discussion now turns to scale-invariant detection, how it is performed and how it is evaluated. The reader will note that it is a straightforward transition from interest point detection. However, the concept of scale and how it is employed to detect regions on an image must first be clarified, which is done next.

3.1 Scale Space

Lindeberg [24] provides a useful, if not formal, concept of scale in the following sentence[1]:

The scale parameter should be interpreted only as an abstract scale parameter implying a weak ordering property of objects of different size without any direct mapping from its actual value to the size of features in a signal represented at that scale.

Scale space is as a multi-scale representation of image information, which is obtained by embedding an image within a family of derived signals that depend on a single parameter: the scale t [24].

Definition 1. *Given a signal* $f : \Re^D \to \Re$, *the linear scale space representation* $L : \Re^D \times \Re \to \Re$ of f *is given by the solution to the diffusion equation*

$$\delta_t L = \frac{1}{2}\nabla^2 L = \frac{1}{2}\sum_{i=1}^{D} \delta_{x_i x_i} L \ , \tag{14}$$

with the initial condition $L(\cdot;0) = f(\cdot)$, *for which the Green function solution is the Gaussian kernel. Equivalently, it is possible to define the scale space as the family of derived signals obtained by convolving a signal f with Gaussian filters at different scales t (standard deviation),*

$$L(\cdot;t) = G_t * f(\cdot) \ . \tag{15}$$

The scale space representation has been cited as a canonical model for biological vision considering the results obtained in neurophysiological studies [39]. Lindeberg [40] defines a *principle for scale selection* that determines the scale at which image features should be analysed.

In the absence of other evidence, assume that a scale level, at which some (possibly non-linear) combination of normalized derivatives assumes a local maximum over scales, can be treated as reflecting a characteristic length of a corresponding structure in the data.

Scale-normalized derivatives are made to be constant over different scales [24], invariant to the scale at which a given feature, such as an edge or corner, is observed; a scale-normalized derivative D of order n is given by

$$D_{i_1,\ldots,i_n}(\mathbf{x};t) = t^n L_{i_1,\ldots,i_n}(\mathbf{x};t) \ . \tag{16}$$

Operators that employ scale-normalized derivatives are said to be scale normalized themselves. Nevertheless, according to Lindeberg, the principal for scale selection ... must be verified empirically, and with respect to the type of problem it is to be

[1] Lindeberg goes on to give a formal derivation of scale space; however, the concepts presented here are sufficient for the present discussion.

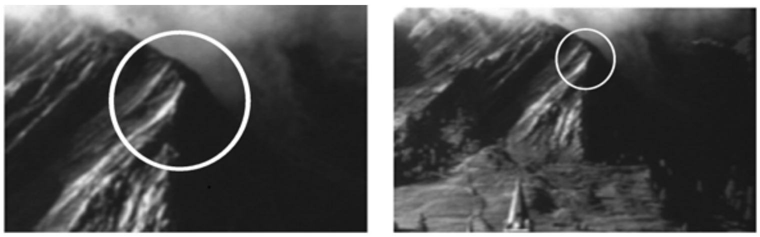


Fig. 8 The same region is detected when viewed at different scales; thus, the scale invariance of the detected regions

applied to. Hence, we can expect that a GP search is a valid approach to construct an interest operator from a *possibly non-linear combination of normalized derivatives* [24].

3.2 Characteristic Scale Selection

From an algorithmic perspective, selecting a characteristic scale for local image features is a process in which local extrema of a function response are found over different scales [40]. Given a function or interest operator $K(\mathbf{x},t_i)$ that computes an interest measure for each image pixel $\mathbf{x}$ at different scales t_i, we can assume that the characteristic scale at $\mathbf{x}$ is t_n if

$$K(\mathbf{x},t_n) > \sup\{K(\mathbf{x_W},t_n),K(\mathbf{x_W},t_{n-1})K(\mathbf{x_W},t_{n+1})|\forall \mathbf{x_W} \in \mathbf{W}, \mathbf{x_W} \neq \mathbf{x}\}$$

$$\wedge\, K(\mathbf{x},t_n) > h\ , \tag{17}$$

where h is a filtering threshold and $\mathbf{W}$ is a $n \times n$ neighbourhood around $\mathbf{x}$. This process, similar to what is done for interest point detection, returns a set of local scale-invariant regions, each centred on an image pixel $\mathbf{x}$ with an associated characteristic scale t_n. Scale invariance is achieved because t_n will vary proportionally with the scale at which the scene feature is observed [17]. This can be observed by defining the size of a given interest region as linearly proportional to its characteristic scale; an example is shown in Fig. 8.

3.3 State-of-the-Art Scale-Invariant Detectors

A brief review of some of the most important literature on scale-invariant feature detection follows. To begin with, Lindeberg [40] presented a scale-adapted Laplacian operator for image feature extraction. Lowe [14] proposed a scale space pyramid where characteristic scale selection was carried out based on local extrema extracted from DoG filters. Mikolajczyk and Schmid [17] gave a comparison of different scale-invariant detectors, including the DoG of Lowe [14] and Lindeberg's [40] Laplacian, along with two detectors they termed the Harris–Laplace and Hessian–Laplace [17]. The Harris–Laplace and Hessian–Laplace schemes detect interest points at different scales, using the scale-adapted Harris and Beaudet operators, respectively, and identify extrema in scale space through a Laplacian operation. Mikolajczyk and Schmid showed, with some experimental results, that their two-stage process can detect regions in a more stable manner than the detectors proposed in [14, 40]. Nonetheless, all the aforementioned scale-invariant detectors work explicitly with the concept of scale space and follow the same basic process of extrema detection given the response of an interest operator, most of which have been used for interest point detection, with the added scale dimension.

Kadir and Brady [41] proposed another scale-invariant detector based on an entropy measure not derived from an explicit scale space analysis. For each image pixel, an intensity histogram is made in a circular region of radius s, the scale. The entropy is computed and local maxima are selected as candidate scales for each region.

The discussion now turns towards describing an implementation of scale-invariant region detectors that are based on the simplest designs found by the GP search described in Sect. 2.3. The operators employed are very compact, especially compared to some of the more elaborate techniques mentioned above, and obtain comparatively high performance scores on standard tests.

3.4 New Scale-Invariant Region Detectors

The choice was made to build two scale-invariant region detectors based on the K_{IPGP1} and K_{IPGP1^*} operators, for two reasons. First, both exhibit the simplest designs found by the GP search process, a feature that can be of important significance due to the additional computational complexity that is introduced with the added scale dimension [17]. Thus, these detectors avoid computationally costly image derivations. Operators with higher computational costs translate poorly to application domains that have strict time requirements, such as vision-based navigation on a mobile robot. Second, both operators gave some of the highest performance scores on interest point detection problems, showing stable and robust detection. Moreover, as stated in Proposition 1, both operators apply DoG filtering, which is a known approximation of the Laplacian operation that is used by the four state-of-the-art detectors presented for comparison [14, 17, 40]. In addition, the additive inverse of K_{IPGP1} is also included in the experimental comparison, hereafter called $-K_{IPGP1}$, with the intention of illustrating the complementary regions both extract, and how K_{IPGP1^*} is related to both. Therefore, the scale-invariant operators, K_{IPGP1}, $-K_{IPGP1}$ and K_{IPGP1^*}, are embedded into the linear scale space representation, such that

$$K_{IPGP1_{t_j}}(\mathbf{x};t) = G_{t_j} * (G_{t_j} * I(\mathbf{x}) - I(\mathbf{x})) \,, \tag{18}$$

$$K_{IPGP1^*_{t_j}}(\mathbf{x};t) = G_{t_j} * |G_{t_j} * I(\mathbf{x}) - I(\mathbf{x})| \,, \tag{19}$$

where $j = 0, 1, ..., M$, and M represents the total number of scales to be analyzed. The operators presented above differ from the original measures presented in Sect. 2.5, for the following reasons. First, for K_{IPGP1}, the two Gaussian's are set to the same scale t_i. It was unclear how to increment the scale parameter of each Gaussian separately in scale space; the important relation that the choice maintains is $\sigma_1 > \sigma_2$ in the DoG filter applied by K_{IPGP1}. Second, for K_{IPGP1^*}, the square of the absolute difference has been omitted because it was experimentally found that it behaved as an intron, lacking any associated fitness gain in scale-invariant detection.

The detection process for the proposed operators is carried out as follows. First, $M = 20$ scales are considered and the scale parameter is such that $t_j = 1.2^j$, which implies a factor of 1.2 between adjacent scales. Therefore, the finest and coarsest scales are given by $t_0 = 1$ and $t_M = 1.2^M$, respectively. However, interest regions are only extracted from scales between $j = 1$ and $j = 20$, because a finer and coarser scale is necessary for characteristic scale detection at scale t_j (see Eq. 17). Hence, the coarsest possible region will have an associated scale $t_n = 38.33$ while the scale of the finest regions will be $t_n = 1.2$.

As mentioned above, [14] also employs a DoG filtering process; nevertheless, important differences exist related to the manner in which scale space is used. In [14], DoG filters are applied between adjacent scales, t_j and t_{j+1}, and each scale response is sub-sampled in order to construct a pyramid representation, which allows a constant size for their filter mask. On the other hand, the detectors implemented

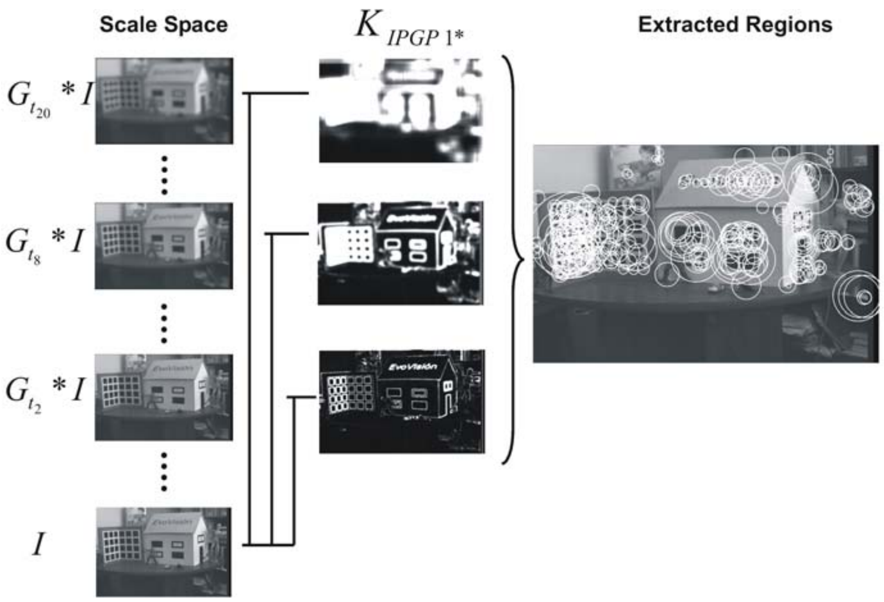


Fig. 9 Scale-invariant region detection with K_{IPGP1^*}. *(Left),* An input image I. *(Middle), interest image* I^* computed at each scale. *(Right),* Detected regions after characteristic scale selection. Regions are drawn with a circle of radius $3 \cdot t_n$

here apply the interest operator using each scale t_j and the initial image I with $t = 0$, which lacks any additional Gaussian smoothing; furthermore, no sub-sampling takes place in scale space. The complete process is depicted graphically in Fig. 9.

3.5 Evaluating Scale-Invariant Region Detectors

Before experimental performance is discussed, a slightly modified repeatability rate must first be described, one that considers the scale change transformations. It is similar to the repeatability rate presented in Sect. 2.2, following the same basic principles with the added difference that now the compared regions will have different sizes as opposed to the constant size used for evaluation of interest points. For this reason, Eq. 3 needs to be augmented before obtaining a repeatability score $r_{I_i}(\varepsilon)$ between an image I_1 and a transformed image I_i. Now, two regions, λ_1 and λ_i, are said to correspond if Eq. 3 holds, where the centre of each region corresponds to x_1 and x_i, and the *overlap error* in the image area that is covered by the regions is below a certain threshold ε_t,

$$\left| 1 - {s_i}^2 \frac{\min(t_1^2, t_i^2)}{\max(t_1^2, t_i^2)} \right| < \varepsilon_t \ , \tag{20}$$

where t_1 and t_i are the associated scales for regions λ_1 and λ_i, s is the actual scale factor recovered from the homography $H_{1,i}$ between images I_1 and I_i [17] and $\varepsilon_t = 0.4$.

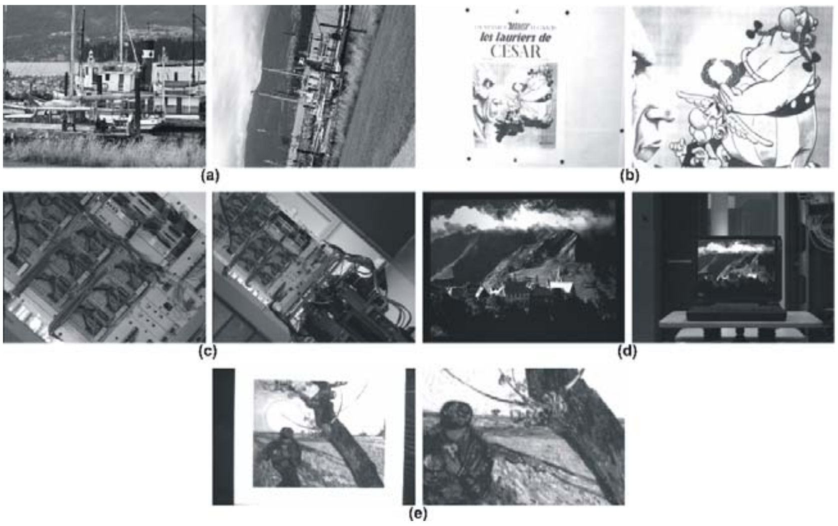


Fig. 10 The reference image (*right) and a transformed image (left) for each testing sequence.* ***(a)*** *Boat (N* $= 5$*, rotation & scale),* ***(b)*** *Asterix (N* $= 16$*, scale),* ***(c)*** *Bip (N* $= 8$*, scale),* ***(d)*** *Laptop (N* $= 20$*, scale), and* ***(e)*** *Van Gogh (N* $= 16$*, scale)*

4 Experimental Evaluation

Figure 11 is a qualitative comparison that shows interest regions extracted by each detector. It is possible to observe how K_{IPGP1} and its additive inverse extract complementary regions while K_{IPGP1^*} extracts a combination of maxima from both. Furthermore, all operators exhibit performance similarities. However, the detectors based on evolved operators are slightly better on most sequences. Figure 12 is a quantitative comparison of each detector; it presents the repeatability rate on five different image sequences (see Fig. 10). The performance graphics plot the images in the sequence and the repeatability rate with respect to the base image. Each

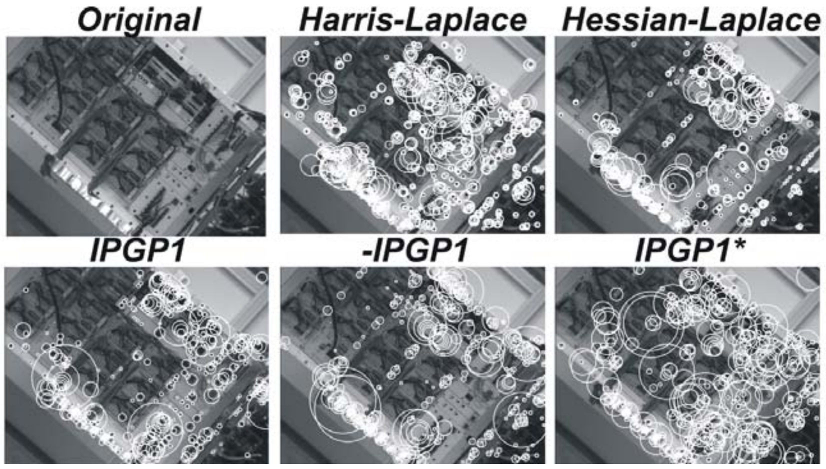


Fig. 11 Sample image regions. Regions are shown with circles of radius $r = 3 \cdot t_n$ pixels, with t_n the regions characteristic scale

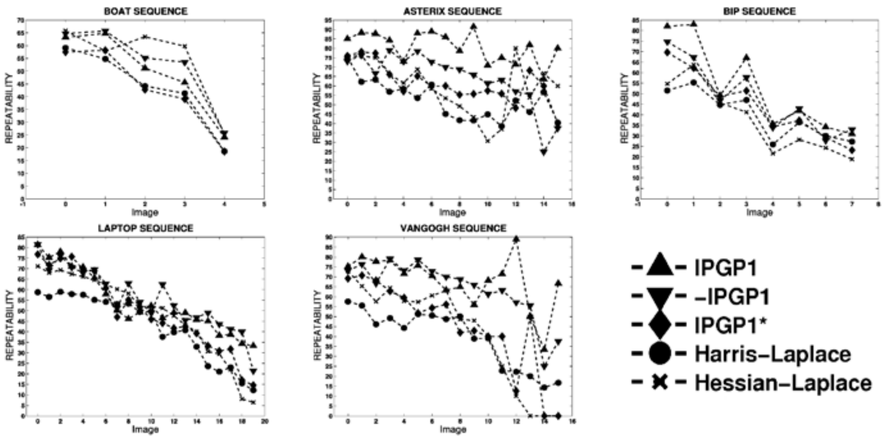


Fig. 12 Repeatability of each detector in the comparison for each test sequence

image in the sequence is progressively transformed. For example, the view point of image 4 is closer to that of the base image than the viewpoint of image 10 in the BIP sequence [38].

5 Concluding Remarks and Future Work

This chapter presented scale-invariant detectors that are based on operators optimized through GP for high repeatability and global dispersion. The detectors were embedded into the linear scale space generated with a Gaussian kernel and compared with state-of-the-art detectors. Results show that the proposed detectors are, on average, better than state-of-the-art detectors based on the repeatability rate. In addition, the proposed operators maintain a simple structure, which is an advantage for real-time applications. This indicates that operators found by simulated evolution can outperform more elaborate manmade designs, a promising result that substantiates the belief that evolution will tend to find the simplest and most apt solution to a given problem. Successful evolution was made possible by correctly framing the search process with a fitness function that promotes the extraction of highly repeatable regions, a property that is useful in many vision applications. As possible future work, two extensions are suggested. First, employ an evolutionary search process that directly accounts for scale space analysis in its fitness evaluation. Special care would be required, however, in order to guarantee a computationally feasible implementation. Second, extend the use of evolved operators to extract affine covariant features, which is a much more challenging problem.

Acknowledgements. Research funded by UC MEXUS-CONACyT Collaborative Research Grant 2005 through the project “Intelligent Robots for the Exploration of Dynamic Environments". First author was supported by scholarship 174785 from CONACyT México. This research was also supported by the LAFMI project, and special thanks are to the Complex Team - INRIA Roquencourt and Grupo de Evolución Artificial - Universidad de Extremadura campus Mérida.

References

1. Marr, D.: Vision: A Computational Investigation into the Human Representation and Processing of Visual Information. Henry Holt and Co., Inc., New York (1982)
2. Kumar, M.P., Torr, P.H.S., Zisserman, A.: Obj cut. In: Proceedings of the 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR), June 20-26, vol. 1, pp. 18–25. IEEE Computer Society, San Diego (2005)
3. Schmid, C., Mohr, R., Bauckhage, C.: Evaluation of interest point detectors. International Journal of Computer Vision 37(2), 151–172 (2000)
4. Oliva, A., Torralba, A.: Modeling the shape of the scene: A holistic representation of the spatial envelope. International Journal of Computer Vision 42(3), 145–175 (2001)

5. Lazebnik, S., Schmid, C., Ponce, J.: Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories. In: Proceedings of the 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR), June 17-22, vol. 2, pp. 2169–2178. IEEE Computer Society, New York (2006)
6. Olague, G., Romero, E., Trujillo, L., Bhanu, B.: Multiclass object recognition based on texture linear genetic programming. In: Giacobini, M. (ed.) EvoWorkshops 2007. LNCS, vol. 4448, pp. 291–300. Springer, Heidelberg (2007)
7. Mikolajczyk, K., Schmid, C.: A performance evaluation of local descriptors. IEEE Transactions on Pattern Analysis and Machine Intelligence 27(10), 1615–1630 (2005)
8. Willamowski, J., Arregui, D., Csurka, G., Dance, C., Fan, L.: Categorizing nine visual classes using local appearance descriptors. In: Proceedings of the 17th International Conference on Pattern Recognition, Workshop Learning for Adaptable Visual Systems, August 23-26. IEEE Computer Society, Cambridge (2004)
9. Sivic, J., Russell, B.C., Efros, A.A., Zisserman, A., Freeman, W.T.: Discovering objects and their localization in images. In: Proceedings of the 10th IEEE International Conference on Computer Vision (ICCV), Beijing, China, October 17-20, vol. 1, pp. 370–377. IEEE Computer Society, Los Alamitos (2005)
10. Trujillo, L., Olague, G., de Vega, F.F., Lutton, E.: Evolutionary feature selection for probabilistic object recognition, novel object detection and object saliency estimation using gmms. In: Proceedings from the 18th British Machine Vision Conference, Warwick, UK, September 10-13, British Machine Vision Association (2007)
11. Harris, C., Stephens, M.: A combined corner and edge detector. In: Proceedings from the Fourth Alvey Vision Conference, vol. 15, pp. 147–151 (1988)
12. Trujillo, L., Olague, G.: Synthesis of interest point detectors through genetic programming. In: Cattolico, M. (ed.) Proceedings of the Genetic and Evolutionary Computation Conference (GECCO), Seattle, Washington, July 8-12, vol. 1, pp. 887–894. ACM, New York (2006)
13. Trujillo, L., Olague, G.: Using evolution to learn how to perform interest point detection. In: Proceedings of the 18th International Conference on Pattern Recognition (ICPR), Hong Kong, China, August 20-24, vol. 1, pp. 211–214. IEEE Computer Society, Los Alamitos (2006)
14. Lowe, D.G.: Object recognition from local scale-invariant features. In: Proceedings of the International Conference on Computer Vision (ICCV), Kerkyra, Corfu, Greece, September 20-25, vol. 2, pp. 1150–1157. IEEE Computer Society, Los Alamitos (1999)
15. Trujillo, L., Olague, G., Legrand, P., Lutton, E.: Regularity based descriptor computed from local image oscillations. Optics Express 15, 6140–6145 (2007)
16. Beaudet, P.R.: Rotational invariant image operators. In: Proceedings of the 4th International Joint Conference on Pattern Recognition (ICPR), Tokyo, Japan, pp. 579–583 (1978)
17. Mikolajczyk, K., Schmid, C.: Scale & affine invariant interest point detectors. International Journal of Computer Vision 60(1), 63–86 (2004)
18. Trujillo, L., Olague, G.: Scale invariance for evolved interest operators. In: Giacobini, M. (ed.) EvoWorkshops 2007. LNCS, vol. 4448, pp. 423–430. Springer, Heidelberg (2007)
19. Mikolajczyk, K., Tuytelaars, T., Schmid, C., Zisserman, A., Matas, J., Schaffalitzky, F., Kadir, T., Van Gool, L.: A comparison of affine region detectors. International Journal of Computer Vision 65(1-2), 43–72 (2005)
20. Koza, J.R.: Genetic Programming: On the Programming of Computers by Means of Natural Selection. MIT Press, Cambridge (1992)

21. Poli, R., Langdon, W.B., McPhee, N.F.: A field guide to genetic programming (2008) (with contributions by J. R. Koza), `http://lulu.com`, `http://www.gp-field-guide.org.uk`
22. Ebner, M.: On the evolution of interest operators using genetic programming. In: Poli, R., et al. (eds.) Late Breaking Papers at EuroGP 1998, pp. 6–10. The University of Birmingham, UK (1998)
23. Ebner, M., Zell, A.: Evolving task specific image operator. In: Poli, R., Voigt, H.-M., Cagnoni, S., Corne, D.W., Smith, G.D., Fogarty, T.C. (eds.) EvoIASP 1999 and EuroEcTel 1999. LNCS, vol. 1596, pp. 74–89. Springer, Heidelberg (1999)
24. Lindeberg, T.: Discrete scale-space theory and the scale-space primal sketch. PhD thesis, Computational Vision and Active Perception Laboratory (CVAP), Royal Institute of Technology, Stockholm, Sweden (1991)
25. Asada, H., Brady, M.: The curvature primal sketch. IEEE Trans. Pattern Anal. Mach. Intell. 8(1), 2–14 (1986)
26. Mokhtarian, F., Suomela, R.: Robust image corner detection through curvature scale space. IEEE Trans. Pattern Anal. Mach. Intell. 20(12), 1376–1381 (1998)
27. Rohr, K.: Recognizing corners by fitting parametric models. Int. J. Comput. Vision 9(3), 213–230 (1992)
28. Olague, G., Hernández, B.: A new accurate and flexible model based multi-corner detector for measurement and recognition. Pattern Recognition Letters 26(1), 27–41 (2005)
29. Moravec, H.P.: Towards automatic visual obstacle avoidance. In: IJCAI, p. 584 (1977)
30. Kitchen, L., Rosenfeld, A.: Gray-level corner detection. Pattern Recognition Letters 1, 95–102 (1982)
31. Wang, H., Brady, J.: Corner detection for 3d vision using array processors. In: Proceedings from BARNAIMAGE 1991, Barcelona, Spain, Secaucus, NJ. Springer, Heidelberg (1991)
32. Förstner, W.: A framework for low level feature extraction. In: Eklundh, J.-O. (ed.) ECCV 1994. LNCS, vol. 801, pp. 383–394. Springer, Heidelberg (1994)
33. Shi, J., Tomasi, C.: Good features to track. In: Proceedings of the 1994 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Seattle, WA, pp. 593–600. IEEE Computer Society, Los Alamitos (1994)
34. Smith, S.M., Brady, J.M.: Susan-a new approach to low level image processing. Int. J. Comput. Vision 23(1), 45–78 (1997)
35. Tissainayagam, P., Suter, D.: Assessing the performance of corner detectors for point feature tracking applications. Image Vision Comput. 22(8), 663–679 (2004)
36. Trujillo, L., Olague, G.: Automated design of image operators that detect interest points. Evolutionary Computation (to appear) (2008)
37. Montana, D.J., Davis, L.: Training feedforward neural networks using genetic algorithms. In: Sridharan, S. (ed.) Proceedings of the Eleventh International Joint Conference on Artificial Intelligence, pp. 762–767. Morgan Kaufman, San Francisco (1989)
38. Visual geometry group, `http://www.robots.ox.ac.uk/vgg/research/`
39. Young, R.A.: Simulation of human retinal function with the gaussian derivative model. In: Proceedings of the 1986 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 564–569. IEEE, Los Alamitos (1986)
40. Lindeberg, T.: Feature detection with automatic scale selection. International Journal of Computer Vision 30(2), 79–116 (1998)
41. Kadir, T., Brady, M.: Saliency, scale and image description. International Journal of Computer Vision 45(2), 83–105 (2001)

Online Evolvable Pattern Recognition Hardware

Kyrre Glette, Jim Torresen, and Moritoshi Yasunaga

Abstract. We propose an evolvable hardware (EHW) architecture for pattern recognition. Compared to an earlier offline EHW pattern recognition approach, the proposed architecture is advantageous in its suitability for online adaptation while maintaining a very high recognition speed. With its support for virtual run-time reconfiguration, the architecture is suitable for implementation in an on-chip evolution system. Function level modules and data buses are employed in the architecture in order to reduce the search space. Furthermore, incremental evolution is applied, which shortens evolution time and allows for the evolution of a larger system. The use of incremental evolution also has the advantage of reducing the size of the evolution hardware in an on-chip evolution system. Variations in architecture parameters are explored, and it is found that the size of the system can be reduced at the cost of longer evolution time or lower recognition accuracy. The architecture is applied to a face image recognition task, for which a recognition accuracy of 96.25% is demonstrated. This result is better than the previously proposed offline EHW architectures.

1 Introduction

Pattern recognition systems requiring a low recognition latency or high throughput could benefit from a hardware implementation. Furthermore, if the systems are applied in time-varying environments, and thus need adaptability, online evolvable hardware (EHW) would seem to be a promising approach [1, 2].

Kyrre Glette and Jim Torresen
University of Oslo, Department of Informatics,
P.O. Box 1080 Blindern, 0316 Oslo, Norway
e-mail: {kyrrehg,jimtoer}@ifi.uio.no

Moritoshi Yasunaga
University of Tsukuba, Graduate School of Systems and Information Engineering, 1-1-1 Ten-ou-dai, Tsukuba, Ibaraki, Japan
e-mail: yasunaga@cs.tsukuba.ac.jp

One approach to online reconfigurability is the virtual reconfigurable circuit (VRC)/virtual field programmable gate array (FPGA) method proposed by Sekanina and Ruzicka [3] and Haddow and Tufte [4]. This method does not change the bitstream to the FPGA itself; rather, it changes the register values of a circuit already implemented on the FPGA, and obtains virtual reconfigurability. This approach has a speed advantage over reconfiguring the FPGA itself, and it is also more feasible because of proprietary formats preventing direct FPGA bitstream manipulation. However, the method requires considerable logic resources.

Experiments on image recognition by EHW were first reported by Iwata et al. [5]. A field programmable logic array (FPLA) device was utilized for recognition of three different patterns from black and white input images of 8$\times$8 pixels. An EHW road image recognition system has been proposed in [6]. A gate array structure was used for categorizing black and white input images with a resolution of 8$\times$4 pixels. Incremental evolution was applied in order to increase the evolvabiliy.

A speed limit sign recognition system has been proposed in [7]. The architecture employed a column of AND gates followed by a column of OR gates, and then a selector unit. A maximum detector then made it possible to decide a speed limit from six categories. Incremental evolution was applied in two ways: each subsystem was first evolved separately, and then in a second step, the subsystems were assembled and the selector units were evolved. The input images were black and white and had a resolution of 7$\times$5 pixels.

An EHW face image classifier system, Logic Design using Evolved Truth Tables (LoDETT), has been presented by Yasunaga et al. [8]. This system is capable of classifying large input vectors into several categories. It has also been successfully applied to sonar spectrum recognition and exon/intron boundary prediction in genome informatics [9]. For a face image recognition task, the input images have a resolution of 8$\times$8 pixels of 8-bit greyscale values, belonging to 40 different categories. In this architecture, the classifier function is directly coded in large AND gates. The classification is based on detecting the category with the highest number of activated AND gates. Incremental evolution is utilized for this system too, where each module for detecting a category is evolved separately. The average recognition accuracy is 94.7% However, evolution is performed *offline* and the final system is synthesized. This approach provides rapid classification in a compact circuit, but lacks run-time reconfigurability.

The system we have developed earlier [10] addresses the reconfigurability by employing a VRC-like array of high-level functions. Online/on-chip evolution is attained, and therefore the system seems suited to applications with changes in the training set. However, the system is limited to recognizing 1 category out of 10 possible input categories. The system uses the same image database as [8] with the same input resolution.

The architecture proposed in this chapter expands to categorization of all 40 categories from the image database used in [8] while maintaining the online evolution features from [10]. The architecture has been changed to accommodate the recognition of multiple categories. While a large number of inputs to the AND gates could be optimized away during circuit synthesis in LoDETT, the run-time reconfiguration aspect of the following architecture has led to a different approach employing fewer elements.

A large amount of literature exists on conventional face image recognition. A comprehensive survey can be found in [11]. Work on conventional hardware face recognition has been undertaken based on the modular PCA method [12]. However, the recognition speed (11 ms) is still inferior to the LoDETT system. Other topics for image processing have also been addresssed using online EHW, such as image filters [13, 14] and image compression [15].

The next section introduces the architecture of the EHW system. Aspects of evolution are discussed in Sect. 3. Results from the experiments are given and discussed in Sects. 4 and 5. Finally, Sect. 6 concludes the chapter.

2 The EHW Architecture

The EHW architecture is implemented as a circuit whose behaviour and connections can be controlled through configuration registers. By writing the genome bitstream from the genetic algorithm (GA) to these registers, one obtains the phenotype circuit which can then be evaluated. This approach is related to the VRC technique as well as to the architectures in our previous works [10, 16].

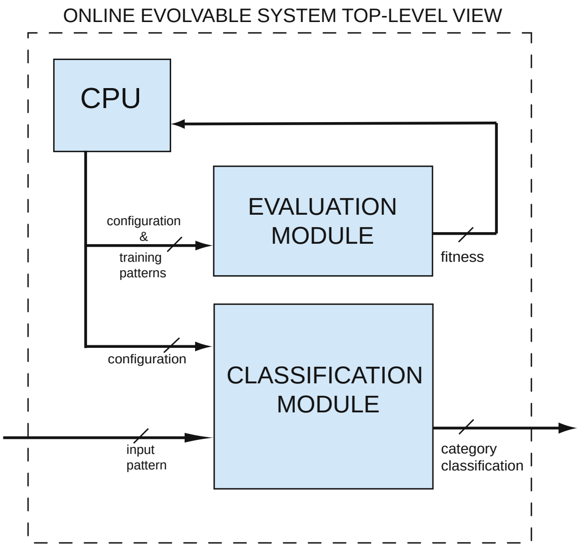


Fig. 1 High-level system view. Classification is performed by the classification module while the CPU and the evaluation module perform evolution of better configurations

2.1 Architecture Overview

A high-level view of the architecture can be seen in Fig. 1. The system consists of three main parts—the classification module, the evaluation module and the CPU. The classification module operates in stand-alone mode except for its reconfiguration, which is carried out by the CPU. In a real-world application, one would imagine some preprocessing module providing the input pattern and possibly some software interpretation of the classification result. The evaluation module operates in close cooperation with the CPU for the evolution of new configurations. The evaluation module accepts a configuration bitstring, also called genome, and calculates the fitness value of the phenotype circuit. This information is in turn used by the CPU for running the rest of the GA. By having a separate evaluation module for evaluation of phenotypes, the classification module can perform classification in parallel with the evolution process. When the entire system is implemented on an FPGA, the CPU will be an embedded processor, such as the PowerPC 405 core in Xilinx Virtex-II Pro or better FPGAs.

2.2 Classification Module Overview

The classifier system consists of K category detection modules (CDMs), one for each category C_i to be classified—see Fig. 2. The input data to be classified are presented to each CDM concurrently on a common input bus. The CDM with the highest output is detected by a maximum detector, and the number of this category will be output from the system. Alternatively, the system could also state the degree of certainty of a particular category by taking the output of the corresponding CDM and dividing by the maximum possible output. In this way, the system could also propose alternative categories in case of doubt.

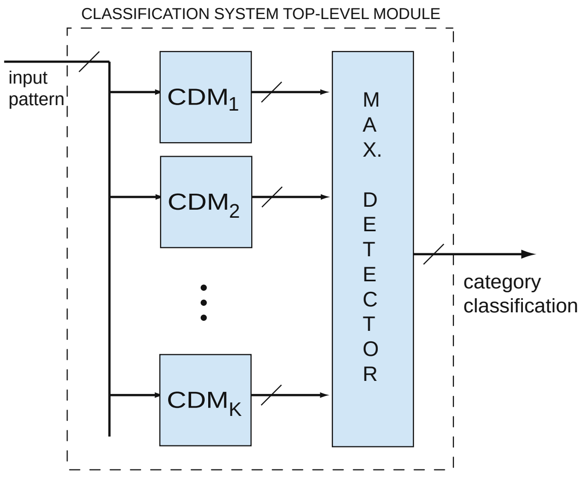


Fig. 2 EHW classifier system view. The pattern to be classified is input to all of the category detection modules

2.3 *Category Detection Module*

Each CDM consists of M “rules" or functional unit (FU) rows (see Fig. 3). Each FU row consists of N FUs. The inputs to the circuit are passed on to the inputs of each FU. The 1-bit outputs from the FUs in a row are fed into an N-input AND gate. This means that all outputs from the FUs must be 1 for a rule to be activated. The outputs from the AND gates are connected to an input counter which counts the number of activated FU rows.

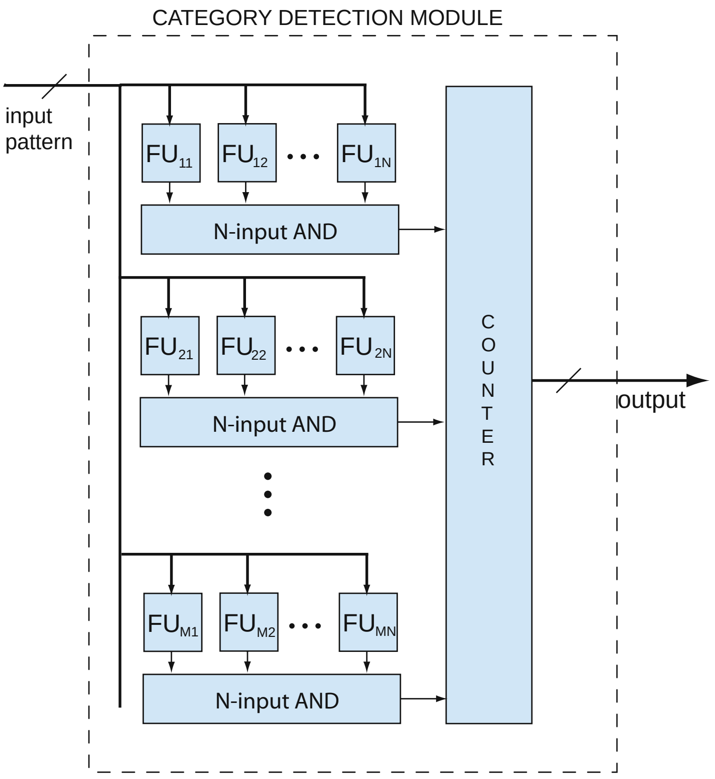


Fig. 3 Category detection module. N functional units are connected to an N-input AND gate. The ouput from M AND gates are connected to an input counter

2.4 *Functional Unit*

The FUs are the reconfigurable elements of the architecture. As seen in Fig. 4, each FU behaviour is controlled by configuration lines connected to the configuration registers. Each FU has all input bits to the system available at its inputs, but only one data element (e.g. one byte) of these bits is chosen. One data element is thus *selected* from the input bits depending on the configuration lines. This data is then fed to the available functions. The choice of functions for this application is detailed

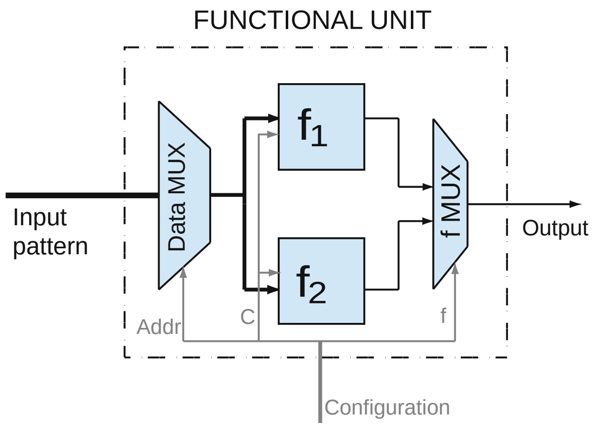


Fig. 4 Functional unit. The configuration lines are shown in grey. The data MUX selects the input data to be fed to the functions f_1 and f_2. The constant C is given by the configuration lines. Finally, the f MUX selects the function that results in output

in Sect. 2.5. In addition, the unit is configured with a constant value C. This value and the input byte are used by the function to compute the output from the unit.

The advantage of selecting which inputs to use is that one is not required to connect to all inputs. A direct implementation of the LoDETT system [8] would have required, in the image recognition case, $N = 64$ FUs in a row. Our system typically uses $N = 6$ units. The rationale is that not all of the inputs are necessary for pattern recognition. This is reflected in the *don't care*s evolved in [8].

2.5 Face Image Recognition

The pattern recognition system has been applied to face image recognition. The fitness of the face recognition system is based on its ability to recognize the correct person from a range of different face images. The images are taken from the AT&T Database of Faces (formerly "The ORL Database of Faces")[1] which contains 400 images divided into 40 people with 10 images each. For each person, images are taken with variations such as different facial expressions and head tilt. The original resolutions of the images were 92×112 pixels, 8-bit greyscale. In our experiment, the images were preprocessed by downsampling to 8×8 pixels, 8-bit greyscale. This was done to reduce noise and the number of inputs to the system. The input pattern to the system is then 64 pixels of 8 bits each (512 bits in total) (see Fig. 5).

Based on the data elements of the input being 8-bit pixels, the functions available to the FU elements have been chosen as *greater than* and *less than*. Through experiments, these functions have been shown to work well, and intuitively, this allows for detecting dark and bright spots. Combined use of these functions for the same pixel makes it possible to define an intensity range. The constant is also 8 bits, and

[1] http://www.cl.cam.ac.uk/Research/DTG/attarchive/facedatabase.html

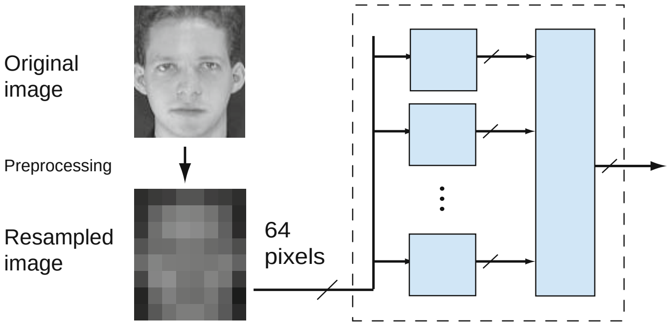


Fig. 5 The original images are resampled to a resolution of 8×8 pixels before being fed to the inputs of the image recognition system

the input is then compared to this value to give *true* or *false* as the output. This can be summarized as follows, with I being the selected input value, O the output and C the constant value:

f	Description	Function
0	Greater than	$O = 1$ if $I > C$, else 0
1	Less than	$O = 1$ if $I < C$, else 0

3 Evolution

This section describes the evolutionary process. The GA implemented for the experiments follows the Simple GA style [17]. The algorithm is written to be run on an embedded processor, such as the PowerPC 405 core [16]. Allowing the GA to run in software instead of implementing it in hardware gives an increased flexibility compared to a hardware implementation.

The GA associates a bit string (genome) with each individual in the population. For each individual, the EHW circuit is configured with the associated bit string, and training vectors are applied on the inputs. By reading back the outputs from the circuit, a fitness value can be calculated.

3.1 Genome

The encoding of each FU in the genome string is as follows:

Pixel address (6 bit)	Function (1 bit)	Constant (8 bit)

This gives a total of $B_{\text{unit}} = 15$ bits for each unit. The genome for one FU row is encoded as follows:

FU_1(15b)	FU_2(15b)	...	FU_N(15b)

The total amount of bits in the genome for one FU row is then, with $N = 8$, $B_{\text{tot}} = B_{\text{unit}} \times N = 15 \times 8 = 120$.

3.2 Incremental Evolution of the Category Detectors

Evolving the whole system in one run would give a very long genome; therefore, an incremental approach is chosen. Each category detector CDM_i is evolved separately, since there is no interdependency between the different categories. This is also true for the FU rows of each CDM. Thus, the evolution can be performed on one FU row at a time. This significantly reduces the genome size.

One then has the possibility of evolving CDM_i in M steps before proceeding to CDM_{i+1}. However, we evolve only *one* FU row in CDM_i before proceeding to CDM_{i+1}. This makes it possible to have a working system in K evolution runs (i.e., $1/M$ of the total evolution time). While the recognition accuracy is reduced with only one FU row for each CDM, the system is operational and improves gradually as more FU rows are added for each CDM.

3.3 Fitness Function

A certain set of the available vectors, V_t, are used for training of the system while the remaining, V_v, are used for verification after the evolution run.

Each row of FUs is fed with the training vectors ($v \in V_t$), and fitness is based on the row's ability to give a positive (1) output for vectors v belonging to its own category ($C_v = C_i$) while giving a negative (0) output for the rest of the vectors ($C_v \neq C_i$).

In the case of a positive output when $C_v = C_i$, the value A is added to the fitness sum. When $C_v \neq C_i$ and the row gives a negative output (value 0), 1 is added to the fitness sum. The other cases do not contribute to the fitness value. The fitness function F for a row can then be expressed as follows way, where o is the output of the FU row:

$$F = \sum_{v \in V_t} x_v \qquad \text{where } x_v = \begin{cases} A \times o \text{ if } C_v = C_i \\ 1 - o \text{ if } C_v \neq C_i \end{cases} \tag{1}$$

For the experiments, a value of $A = 64$ has been used. This emphasis on the positive matches for C_i has shown to speed up the evolution.

3.4 Evolution Parameters

For the evolution, a population size of 30 is used. Elitism is used; thus, the best individual from each generation is carried over to the next generation. The (single point) crossover rate is 0.9, and therefore the cloning rate is 0.1. A roulette wheel selection scheme is applied. Linear fitness scaling is used with six expected copies of the best individual. The mutation rate is expressed as a probability for a certain number, n, of mutations on each genome. The probabilities are as follows:

n	1	2	3	4
$p(n)$	$\frac{7}{10}$	$\frac{1}{10}$	$\frac{1}{10}$	$\frac{1}{10}$

4 Results

This section presents the results of the experiments undertaken. The results are based on a software simulation of the EHW architecture.

4.1 Architecture Parameters

The architecture parameters, N and M, i.e., the number of FUs in an FU row and the number of FU rows in a CDM, respectively, have been evaluated. As can be seen in Fig. 6, the number of generations required to evolve an FU row is dependent on the number of FUs in such a row.

A configuration with few FUs makes it difficult to find an FU row distinguishing between the right and wrong categories, thus increasing evolution time. Configurations with many FUs also suffer from increased evolution time; these configurations have the problem of a longer genome and thus a larger search space. For the current application, rows of N=8 FUs are the easiest to evolve, but maximum fitness values have been evolved for FU rows with values of N down to 4. We have not seen any discernable connection between the number of FUs in an FU row and the recognition accuracy as long as the row could be evolved to a maximum fitness value.

However, increasing the number of FU rows for a category leads to an increase in the recognition accuracy, as seen in Fig. 7. As the number of FU rows increases, so does the output resolution from each CDM. Each FU row is evolved from an initial random bitstream, which ensures a variation in the evolved FU rows.

4.2 Recognition Accuracy

Ten evolution runs were conducted, each with a different selection of test vectors. That is, the 10% of the images used as test vectors were chosen differently in each evolution run. For $K = 40$, $M = 8$ and $N = 6$, an average recognition accuracy of 96.25% has been achieved. This result is slightly better than the average accuracy of 94.7% reported from the LoDETT system [8].

An experiment with varying number of training vectors has also been undertaken. one to nine training vectors were used for each category. The rest were used as test

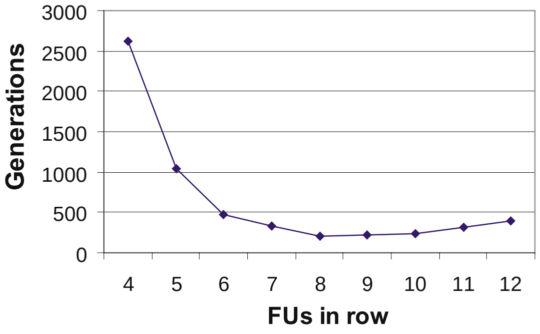


Fig. 6 Generations required to evolve rows of different number of FUs to maximum fitness. Average over five evolution runs

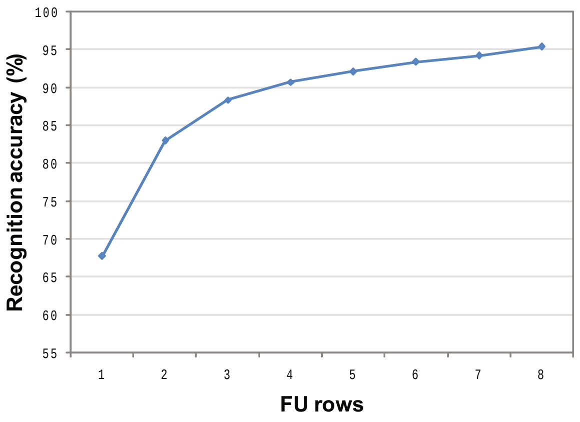


Fig. 7 Accuracy obtained for varying number of FU rows. A fixed row size of 8 was used. Average over 20 runs

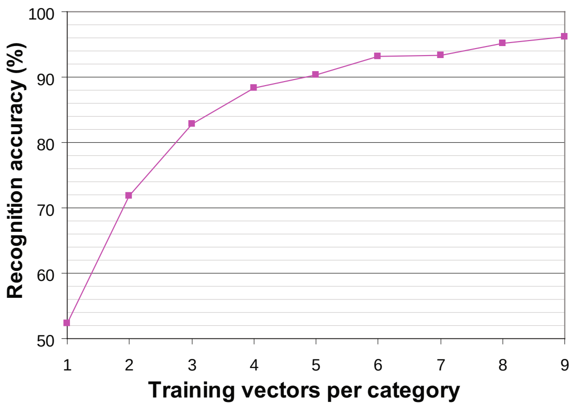


Fig. 8 Accuracy obtained for varying number of training vectors per category, with $N = 6$ and $M = 10$

vectors for calculating the accuracy. The results can be seen in Fig. 8. The results are competitive to traditional image recognition algorithms' results on the same dataset, such as Eigenfaces (around 90.0% for eight training vectors per category) [18] or Fisherfaces (around 95.0% for eight training vectors) [18], but other methods, such as KNR-based eigenspectra (97.8%) [19] or SVM (98%) [20], perform better.

4.3 Evolution Speed

For $K = 40$, $M = 6$ and $N = 10$, the average number of generations (over 10 runs) required for each evolution run (i.e., one FU row) is 219; thus, an average of 52,560 generations is required for the entire system. The average evolution time for the system is 140 s on an Intel Xeon 5160 processor using 1 core. This gives an average of 0.6 s for one FU row or 23.3 s for 40 rows (the time before the system is operational). It is expected that a hardware implementation will yield lower training times, as the evaluation time for each individual will be reduced.

4.4 Hardware Implementation

A preliminary implementation of an FU row has been carried out for an FPGA in order to achieve an impression of resource usage. When synthesized for a Xilinx XC2VP30, eight FU rows of six FUs (for one CDM) give the following results:

Resource	Used	Available	Percentage
4-input LUTs	598	27,392	2
Slices	328	13,696	2

In this case, the data selector MUX in the FU is implemented by using time multiplexing, since directly implementing a 64$\times$8-bit multiplexer for each FU requires many resources in the FPGA. The downside of time multiplexing is that each of the 64 pixels must be present on the inputs for a clock cycle before the classification can be made. With an estimate of maximum 10 cycles needed for the bit counter and the maximum detector, the system would require 74 clock cycles in order to classify a pattern. For a 100-MHz system, this would give more than 1 M classifications per second, i.e., less than 1μs for one image.

5 Discussion

This section discusses the results and features of the proposed architecture as well as possible future modifications.

5.1 Online Evolution

The main improvement of this system over the LoDETT system is the aspect of online evolution. This is achieved by adapting a VRC-like architecture, allowing for quick reconfiguration. A naive implementation of LoDETT for online evolution would have posed a problem because of the large AND gates (1024 inputs in the face image recognition case). The offline approach of LoDETT makes it possible to reduce the final FPGA circuit size by eliminating unused (*don't care*) AND gate inputs during circuit synthesis, but this is not possible with an online approach. However, by selecting a subset of the input pixels for each image, the size of the circuit can be kept down. The drawback of this method is the extra cycles or FPGA resources needed for implementing the pixel selector MUXes. A positive side effect of the architecture change, compared to LoDETT, is the increased recognition accuracy.

5.2 Generalization and System Size

The system seems to be able to generalize better as more FU rows are added for each CDM. There is nothing that prevents these new rows from evolving into the same as the other rows, but the random initialization of the genome (and thus the FU row), combined with several possible solutions giving maximum fitness, seems to ensure a diversity, which is useful for generalization. This differs from the LoDETT system where each row is directly related to a training vector, and *don't cares* are evolved for generalization.

Because of the tradeoff between classification accuracy (by using more FU rows) and HW resources, these parameters should be adapted according to the resources of the target HW platform. Moreover, the size of the system can be reduced by employing fewer FUs per FU row, at the cost of increased evolution time.

5.3 Real-Time Adaptation

Real-time adaptation can be achieved by having evolution running on one separate FU row in the evaluation module, implemented on the same chip as the operational classifier system. Thus, when there are changes in the training set (e.g. new samples are added), the FU rows can be re-evolved and replaced, one at a time, while the main classification module is operational using the currently best configuration. Since one FU row requires few hardware resources compared to the full classification module, little overhead is added.

5.4 Future Work

A full system-on-chip hardware implementation is planned. The GA will run on an embedded processor in a Xilinx FPGA. Evolution speed as well as recognition speed should be measured. Furthermore, it would be interesting to improve the architecture by dynamically adjusting the number of FUs and rows for each CDM, depending on its evolvability. Other architectural changes could also be considered, e.g. some kind of hierarchical approaches, for increased recognition accuracy or reduced resource usage. Partial reconfiguration could also be explored as a means of reducing the circuit size.

Second, variations of the architecture should be tested on other pattern recognition problems. Since the main advantage of this architecture is its very high recognition speed, applications requiring high throughput should be suitable. The LoDETT system has been successfully applied to genome informatics and other applications [21, 22]. It is expected that the proposed architecture also could perform well on similar problems, if suitable functions for the FUs are found.

6 Conclusions

An EHW architecture for a complex pattern recognition task has been proposed. The architecture supports run-time reconfiguration and is thus suitable for implementation in an on-chip evolution system. The proposed architecture utilizes data buses and higher-level functions in order to reduce the search space. In addition, evolution of the system follows an incremental approach. Short evolution time is needed for a basic working system to be operational. Increased generalization can then gradually be added. System size can be reduced at the cost of increased evolution time or lower recognition accuracy. The classification accuracy has been shown to be slightly better than earlier offline EHW approaches. The system seems suitable for applications requiring high speed and online adaptation to a changing training set.

Acknowledgements. This research is funded by the Research Council of Norway through the project *Biological-Inspired Design of Systems for Complex Real-World Applications* (proj. no. 160308/V30).

References

1. Yao, X., Higuchi, T.: Promises and challenges of evolvable hardware. In: Higuchi, T., Iwata, M., Weixin, L. (eds.) ICES 1996. LNCS, vol. 1259, pp. 55–78. Springer, Heidelberg (1997)
2. Torresen, J.: Possibilities and limitations of applying evolvable hardware to real-world application. In: Grünbacher, H., Hartenstein, R.W. (eds.) FPL 2000. LNCS, vol. 1896, pp. 230–239. Springer, Heidelberg (2000)
3. Sekanina, L., Ruzicka, R.: Design of the Special Fast Reconfigurable Chip Using Common FPGA. In: Proceedings of the IEEE Conference on Design and Diagnostics of Electronic Circuits and Systems (DDECS), pp. 161–168 (2000)
4. Haddow, P., Tufte, G.: Bridging the genotype-phenotype mapping for digital FPGAs. In: Proc. of the Second NASA/DoD Workshop on Evolvable Hardware (2001)
5. Iwata, M., Kajitani, I., Yamada, H., Iba, H., Higuchi, T.: A pattern recognition system using evolvable hardware. In: Ebeling, W., Rechenberg, I., Voigt, H.-M., Schwefel, H.-P. (eds.) PPSN 1996. LNCS, vol. 1141, pp. 761–770. Springer, Heidelberg (1996)
6. Torresen, J.: Scalable evolvable hardware applied to road image recognition. In: Lohn, J. (ed.) Proc. of the 2nd NASA/DoD Workshop on Evolvable Hardware, pp. 245–252. IEEE Computer Society, Silicon Valley (2000)
7. Torresen, J., Bakke, W.J., Sekanina, L.: Recognizing speed limit sign numbers by evolvable hardware. In: Yao, X., Burke, E.K., Lozano, J.A., Smith, J., Merelo-Guervós, J.J., Bullinaria, J.A., Rowe, J.E., Tiňo, P., Kabán, A., Schwefel, H.-P. (eds.) PPSN 2004. LNCS, vol. 3242, pp. 682–691. Springer, Heidelberg (2004)
8. Yasunaga, M., Nakamura, T., Yoshihara, I., Kim, J.: Genetic Algorithm-based Design Methodology for Pattern Recognition Hardware. In: Miller, J.F., Thompson, A., Thompson, P., Fogarty, T.C. (eds.) ICES 2000. LNCS, vol. 1801, pp. 264–273. Springer, Heidelberg (2000)
9. Yasunaga, M.: et al.: Evolvable reasoning hardware: Its application to the genome informatics. In: Proc. of IEEE Congress on Evolutionary Computation, pp. 704–711 (2001)
10. Glette, K., Torresen, J., Yasunaga, M., Yamaguchi, Y.: On-Chip Evolution Using a Soft Processor Core Applied to Image Recognition. In: Proceedings 1st NASA /ESA Conference on Adaptive Hardware and Systems (AHS), pp. 373–380. IEEE CS Press, Los Alamitos (2006)

11. Zhao, W., Chellappa, R., Phillips, P.J., Rosenfeld, A.: Face recognition: A literature survey. ACM Comput. Surv. 35(4), 399–458 (2003)
12. Ngo, H., Gottumukkal, R., Asari, V.: A flexible and efficient hardware architecture for real-time face recognition based on eigenface. In: Proc. of IEEE Computer Society Annual Symposium on VLSI, pp. 280–281. IEEE, Los Alamitos (2005)
13. Sekanina, L.: Virtual reconfigurable circuits for real-world applications of evolvable hardware. In: Tyrrell, A.M., Haddow, P.C., Torresen, J. (eds.) ICES 2003. LNCS, vol. 2606, pp. 186–197. Springer, Heidelberg (2003)
14. Zhang, Y., Smith, S.L., Tyrrell, A.M.: Digital circuit design using intrinsic evolvable hardware. In: Evolvable Hardware, pp. 55–62. IEEE Computer Society Press, Los Alamitos (2004)
15. Tanaka, M., Sakanashi, H., Salami, M., Iwata, M., Kurita, T., Higuchi, T.: Data compression for digital color electrophotographic printer with evolvable hardware. In: Sipper, M., Mange, D., Pérez-Uribe, A. (eds.) ICES 1998. LNCS, vol. 1478, pp. 106–114. Springer, Heidelberg (1998)
16. Glette, K., Torresen, J.: A flexible on-chip evolution system implemented on a Xilinx Virtex-II Pro device. In: Moreno, J.M., Madrenas, J., Cosp, J. (eds.) ICES 2005. LNCS, vol. 3637, pp. 66–75. Springer, Heidelberg (2005)
17. Goldberg, D.: Genetic Algorithms in search, optimization, and machine learning. Addison-Wesley, Reading (1989)
18. Zhou, D., Yang, X.: Face recognition using enhanced fisher linear discriminant model with facial combined feature. In: Zhang, C., W. Guesgen, H., Yeap, W.-K. (eds.) PRICAI 2004. LNCS, vol. 3157, pp. 769–777. Springer, Heidelberg (2004)
19. Liu, B., Zhang, J.: Eigenspectra versus eigenfaces: Classification with a kernel-based nonlinear representor. In: Wang, L., Chen, K., S. Ong, Y. (eds.) ICNC 2005. LNCS, vol. 3610, pp. 660–663. Springer, Heidelberg (2005)
20. Kim, K., Kim, J., Jung, K.: Recognition of facial images using support vector machines. In: Proceedings of the 11th IEEE Signal Processing Workshop on Statistical Signal Processing, pp. 468–471. IEEE, Los Alamitos (2001)
21. Yasunaga, M., et al.: Gene finding using evolvable reasoning hardware. In: Tyrrell, A.M., Haddow, P.C., Torresen, J. (eds.) ICES 2003. LNCS, vol. 2606, pp. 198–207. Springer, Heidelberg (2003)
22. Yasunaga, M., Kim, J.H., Yoshihara, I.: The application of genetic algorithms to the design of reconfigurable reasoning vlsi chips. In: FPGA 2000: Proceedings of the 2000 ACM/SIGDA eighth international symposium on Field programmable gate arrays, pp. 116–125. ACM Press, New York (2000)

A Variant Program Structure in Tree-Based Genetic Programming for Multiclass Object Classification

Mengjie Zhang and Mark Johnston

Abstract. This chapter describes an approach to the use of genetic programming for multiclass object classification. Instead of using the standard tree-based genetic programming approach, where each genetic program returns just one floating point number that is then translated into different class labels, this approach invents a new program structure with multiple outputs, each for a particular class. A voting scheme is then applied to these output values to determine the class of the input object. The approach is examined and compared with the standard genetic programming approach on four multiclass object classification tasks with increasing difficulty. The results show that the new approach outperforms the basic approach on these problems. A characteristic of the proposed program structure is that it can easily produce multiple outputs for multiclass object classification problems, while still keeping the advantages of the standard genetic programming approach for easy crossover and mutation. This approach can solve a multiclass object recognition problem using a single evolved program in a single run.

1 Introduction

Object classification arises in a very wide range of applications, such as detecting faces from video images, recognizing digits from postal codes and diagnozing all tumours in a database of x-ray images. In these classification problems, there are usually three or more classes of examples to be distinguished. Clearly, performing these tasks manually is too expensive or too slow. Given the amount of image data that needs to be classified, computer-based solutions to many of these tasks would be of immense social and economic value.

A solution to an object classification task would involve a program that can correctly map an input vector describing an object image instance (such as a face image,

Mengjie Zhang and Mark Johnston
School of Mathematics, Statistics and Computer Science, Victoria University of Wellington, P.O. Box 600, Wellington, New Zealand
e-mail: `mengjie.zhang,mark.johnston@ecs.vuw.ac.nz`

a coin image or a fingerprint image) to one of a small set of class labels. Manually writing such computer programs is difficult, time consuming and often infeasible: human programmers are often unable to identify all the subtle and interrelated conditions that are needed to distinguish between the different classes.

Derived from genetic algorithms, evolutionary programming, machine learning and automatic programming [1–3], genetic programming (GP) is a fast developing evolutionary learning and search paradigm [4–6]. Different from traditional learning and search methods such as hill climbing, gradient descent search and simulated annealing search [7], which usually take more advantage of the details of an individual creature and hence learn by the individual changes, the evolutionary learning process in GP learns by following a genetic beam search to produce a set of skilful survivors though these creatures do not themselves learn during their individual lifetime. In GP, solutions to a problem can be represented in different forms but are usually interpreted as computer programs. Darwinian principles of natural selection and recombination are used to evolve a population of programs towards an effective solution to specific problems. The flexibility and expressiveness of computer program representation, combined with the powerful capabilities of evolutionary search, make GP an exciting method to solve a great variety of problems. A strength of this approach is that evolved programs can be much more flexible than the highly constrained, parameterized models used in other techniques such as neural networks and support vector machines.

Since the early 1990s, a variety of GP systems with different representations have been developed. These include tree-based GP [4, 5], linear GP [8], graph-based GP [9, 10], linear-graph GP [11], grammar-based GP [12] and even machine code instruction-based GP [13]. Among them, tree-based or tree-like GP is one of the most commonly used representations and has been considered the standard GP method. While GP with this representation has been widely applied to producing expressions for symbolic regression problems, it has also been considered a promising approach for building reliable classification programs quickly and automatically, given only a set of example data on which a program can be evaluated. It has been successfully applied to real-world classification problems such as detecting and recognizing particular classes of objects in images [14–21], demonstrating the potential of GP as a general method to solve classification problems.

The standard tree-based GP evolves programs that map a vector of input values to a single real-valued output [4, 22–25]. For object classification tasks, this output must be mapped or translated into a set of class labels. For binary object classification problems, there is a natural translation of negative values to one class and positive values to the other class, which is relatively straightforward. For multiclass object classification problems, finding the appropriate region boundaries on the numeric value to separate the different classes well is much more difficult. Current solutions generally separate the program output space into distinct regions for different classes. These include a primary static method such as *program classification map* [24] or *static range selection* [25], and more complicated *dynamic range selection* [25] and *centred and slotted dynamic class boundary determination* [26].

Past works have demonstrated the effectiveness of these approaches for a number of object classification problems [27, 28].

In static methods, the program's output space needs to be manually partitioned into fixed regions in a fixed order, resulting in either the expense of human experts on the task or inappropriate partitioning and hence poor performance. In dynamic methods, class boundaries can be automatically learnt or evolved, but they require much larger search space, which often results in unnecessarily complex programs [19, 26].

An alternative approach is to decompose a multiclass object classification problem into multiple binary object classification tasks first, and then apply GP to each binary object classification task [25]. While this approach can successfully avoid the problems of the multiple separate regions mentioned above, it has to evolve multiple programs or use multiple GP runs for a single multiclass object classification task. For a new unseen object image instance, the multiple evolved programs must be used together to predict which class this object image belongs to. Accordingly, the evolutionary training time and execution time of this approach could be quite long and often cannot meet the requirements of real-time applications.

Another alternative approach would be to use strongly typed GP in which the evolved programs return values of different data types [29]. However, evolving such programs is difficult because of the increased complexity of the set of operators and terminals required.

1.1 Goals

To avoid the problems described above, the goal of this chapter is to investigate a new program structure for multiclass object classification. Rather than using the standard tree structure to represent the evolved programs, each of which only returns a single floating point value from the root node that is then translated into different class labels, the proposed program structure will be able to produce multiple outputs, each corresponding to a class. The class label is then determined by simple voting, thus avoiding the painful class label translation step. On the other hand, we want the new structure/representation to keep (at least partially keep) the "shape" of the tree structure so that the advantages of the standard GP can be retained, for examples, the crossover and mutation operators can still be easily performed. To test the performance of this new structure, the GP system with this new program structure will be examined and compared with the standard GP system having the standard tree representation on a sequence of multiclass object classification problems of increasing difficulty.

1.2 Organization

The rest of the chapter is organized as follows. Section 2 briefly describes the background and related work. Section 3 describes the new program structure and related issues. Section 4 describes the image data sets for multiclass object classification tasks and experiment configurations. Section 5 presents the results with discussions. Section 6 draws the conclusions and gives future working directions.

2 Background

This section first describes the field of object recognition, then briefly reviews previous work on GP related to object classification and recognition, and finally overviews the class translation rule commonly used in the standard GP approach to multiclass object classification problems.

2.1 Object Classification and Recognition

Object recognition, also known as *automatic object recognition* or *automatic target recognition*, is a specific field and a challenging problem in computer vision and image understanding [30].

Traditionally, most research on object recognition involves four stages: *preprocessing, segmentation, feature extraction* and *classification* [31, 32]. The preprocessing stage aims to remove noise or enhance edges of the possible regions or objects in the input images. In the segmentation stage, a number of coherent regions and "suspicious" regions which might contain objects are usually located and separated from the entire images. The feature extraction stage extracts domain-specific features from the segmented regions. Finally, the classification stage uses these features to distinguish the classes of the objects of interest (separated regions). The features extracted from the images and objects are generally domain specific such as high-level relational image features. Data mining and machine learning algorithms are usually applied at the classification stage.

2.2 GP-Related Work to Object Classification and Recognition

Object recognition has been of tremendous importance in many application domains. Since the early 1990s, GP has been used for object classification and recognition in almost all of the application domains. These include military applications such as vehicle detection [15, 17, 33], human face recognition [20, 34], human eye and mouth recognition and detection [35, 36], character/letter recognition [5, 37], medical image classification and detection [19, 25], orthodontic landmark detection [38], shape and coin recognition and detection [18, 26], texture classification [22, 39] and other object detection and recognition [40, 41].

Due to space constraints, we cannot describe this big area in detail. More details of this area can be seen from a recent book [42], a recent special journal issue [43], recent EvoIASP proceedings and the special session on Evolutionary Computer Vision on IEEE Congress on Evolutionary Computation in recent years.

2.3 GP for Multiclass Object Classification: Class Translation Rule

As we will need to compare the GP approach with the new program structure to the basic GP approach, we now briefly review a common class translation rule used for multiclass object classification. This class translation rule is used to convert a single floating point number output from the root of an evolved program into a class label.

In the basic GP approach, we use a variant program classification map [24] or static range selection method [25], as the class translation rule. In this method, two or more pre-defined thresholds/boundaries are applied to the numeric output value of the genetic program and the ranges/regions between these boundaries are translated into different classes. This method is simple because these regions are set by the fixed boundaries at the beginning of evolution and remain constant during evolution.

If there are n classes in a classification task, these classes are sequentially assigned n regions along the numeric output value space from some negative numbers to positive numbers by $n-1$ thresholds/boundaries. Class 1 is allocated to the region with all numbers less than the first boundary, class 2 is allocated to all numbers between the first and the second boundaries and class n to the region with all numbers greater than the last boundary. More formally, if we use v to represent an input feature vector for an instance in the classification problem, and p(v) to represent the output value of an evolved genetic program p, then the class of the instance in the data set classified by program p will be determined as Eq.1.

$$\text{class}(p,v) = \begin{cases} \text{Class } 1, & p(v) < T_1 \\ \text{Class } 2, & T_1 \le p(v) < T_2 \\ \ldots\ldots & \ldots\ldots \\ \text{Class } i, & T_{i-1} \le p(v) < T_i \\ \ldots\ldots & \ldots\ldots \\ \text{Class } n, & T_{n-1} \le p(v) \end{cases} \tag{1}$$

where the output value of program p is a floating point number, and T_1, T_2, ..., T_{n-1} are static, pre-defined class boundaries (thresholds) along the floating point number space.

Although this class translation rule is easy to set and has achieved some success in several problems [19, 27], it has a number of disadvantages. First, the ordering of classes is fixed. For binary classification problems, we only need one boundary value (usually zero) to separate the program output space into two regions, which is quite reasonable. For multiple class problems, fixing the ordering of different classes is clearly not good in many cases, since it is very difficult to set a good ordering of different classes without sufficient prior domain knowledge. Second, the regions along the program output space are also fixed for these classes. In this way, we need to determine a number of additional parameters for the evolution. In practice, this is also hard and often needs prior knowledge or empirical tuning.

3 New Program Structure and Representation

We developed a new program structure/representation for multiclass object classification problems. In the rest of this section, we will describe program structure, evaluation, representation, program generation, related genetic operators and a summary of characteristics of the new program structure.

3.1 Program Structure

The genetic programs in a population use a new program structure consisting of two main parts: (a) a modified program tree structure (for presentation convenience, we called it *MPT*) and (b) an associated *output vector* for holding outputs from the MPT program, as shown in Fig. 1.

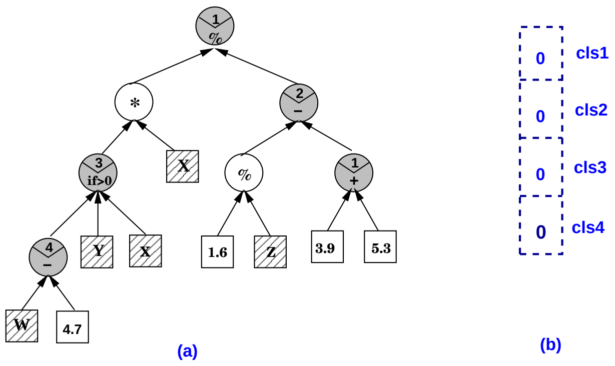


Fig. 1 An example of the new program structure. (a) Modified program tree (MPT); (b) Output vector

Similar to the standard program tree structure, the MPT also has a root node, multiple internal nodes and multiple leaf nodes. Feature terminals (*hatched squares*) and random constants (*outlined squares*) form the leaf nodes. Operations in the function set form the root and internal nodes (*shaded circles and outlined circles*).

Internal nodes in MPTs are not always the same as in the standard program tree structure. In an MPT, the internal nodes are categorized into two kinds. The first kind is the normal function nodes as in the standard program tree, as shown in outlined circles in Fig. 1(a), and the other kind of nodes are those that include *special function structure*, as shown in grey filled circles in Fig. 1(a). For presentation convenience, we call these special function nodes *modi* nodes.

In addition to the normal function operators in the function set, each modi node also has a number in the node structure, which is associated with the corresponding class label, for example, number 1 for *class 1*. In this way, the subtree below the modi node (whose root is the modi node) will be "structurally connected" to the element in the output vector in Fig. 1(b) corresponding to that class. Accordingly, a modi node has two roles. The first role is that it *updates* an element in the output vector that the node is associated with by adding the value of the subtree with the modi node as the root to the original value of the element in the output vector. In this way, unlike the standard program tree structure, which outputs just one floating point number through the root, an MPT program connects to the *output vector* and produces multiple values, each of which corresponds to a single class in multiclass object classification problems.

To maintain the completeness of the MPT as in the standard program tree structure, the second role of a modi node is that it passes the return value of its left child node to its parent. In this way, the entire MPT program tree will not be broken by the modi nodes but can still be regarded as a normal program tree for evaluation. This is also important as we want the new MPT structure to still benefit from the advantages of the standard program tree structure in GP. Another consideration of allowing a modi node to pass its left child to its parent is the reuse of that child node, which will be discussed in more detail later in this section.

In the new structure, the output vector is considered *virtual* (shown as dashed in Fig. 1 b). It does not physically “exist” during the whole evolutionary process but only at the moment when the program is being evaluated. During evolution, the output vectors of all programs “disappear”; only the MPTs are active and take part in evolution. Only at the program evaluation time is the output vector realized and receives update from the MPT program trees via the modi nodes.

3.2 Evaluation of MPT Programs

Figure 2 shows the evaluation process of the MPT programs. When the evaluation starts, the virtual output vector is realized and all the elements of the vector are initialized to zero. As shown in Fig. 2, the dashed vector becomes solid. During the evaluation process, each *non*-modi function node returns the value of the subtree at the node to its parent node, exactly the same as in the standard program tree. The modi nodes do two things. Each modi node first uses the value of the subtree at the node to update the output vector, then passes on the value of its left child to its parent node. The consequence of the program evaluation is that the element values of the output vector are updated by all the modi nodes in the MPT tree during the evaluation process based on the input object. At the end of the evaluation process, multiple floating point numbers in the output vector are produced, each of which corresponds to a class. A voting strategy (winner takes all) is then applied to those outputs and the winner class (the one with the maximum value) is considered to be the class of the input object.

Note that the connections of the modi nodes with their parent nodes have been changed to a fine dotted line in Fig. 2, as those “connections” do not actually exist during the program evaluation process—the modi subtrees only return the value of their left child subtrees to their parents.

As an example, consider a four class object classification task with class labels `{cls1, cls2, cls3 and cls4}` and an object to be classified, represented

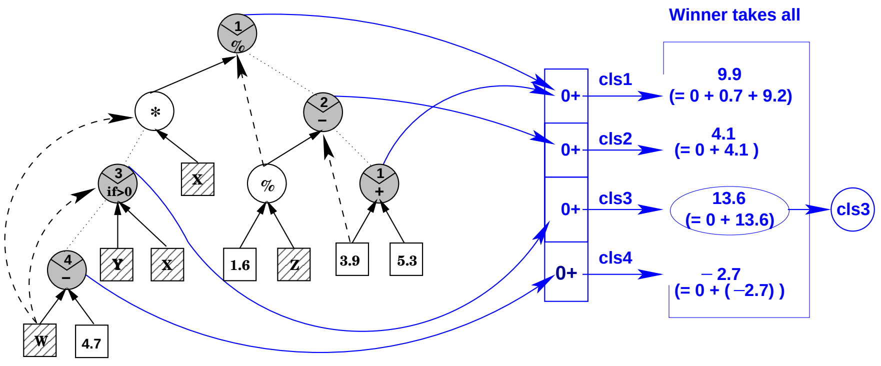


Fig. 2 Evaluation of the example Modi program structure

by an input vector that consists of six extracted feature values `[V,U,W,X,Y,Z]=` `[0.6,5.7,2.0,2.8,13.6, 0.2]`. Assuming the MPT program in Fig. 2 is the evolved/learnt classifier, feeding the input vector into the classifier would modify/update the output vector to `[9.9, 4.1, 13.6, -2.7]`, as shown in Fig. 2 (right). In this case, this object is classified into the third class `cls3`, as the third output `13.6` is the *winner*.

3.3 DAG Representation of MPT Programs

To reveal the high-level representation of the MPT programs with the new structure, we further analyze the program evaluation process. If we remove the fine dotted lines in Fig. 2 as they do not do anything during evaluation, then Fig. 2 can be rewritten as Fig. 3.

As can be seen from Fig. 3, the MPT programs, during dynamic evaluation, represent a kind of directed acyclic graph (DAG), although the real structure of the MPT programs just uses a tree and a vector. Therefore, it is not surprising that the

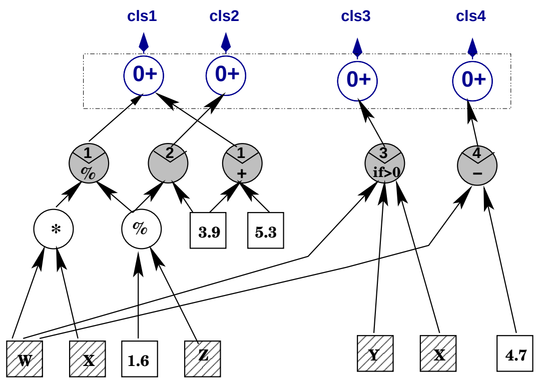


Fig. 3 Modi simulated graph/network classifier

MPT structure can naturally do multiclass object classification—the DAG representation allows the MPT structure to easily produce multiple output values, each for a class, just like multilayer feedforward neural networks [44].

3.4 Program Generation and Modi Rate μ

The generation method of the MPT program trees is very similar to that of the standard program tree, except for three additional rules for generating the modi nodes in MPT programs. First, all leaf nodes should not be set to modi nodes since a modi node requires a left child to return to its parent node. Second, the root node is always set to a modi node in order to guarantee that no part of the MPT tree is wasted. Finally, for internal nodes, the probability of a node to be set to modi is controlled by the *modi rate*, $\mu \in [0,1]$, defined by the user. The modi rate can be interpreted as the expected proportion of modi nodes over all internal nodes in programs of the *initial* population. Element indices of the output vector are then assigned randomly and uniformly across all modi nodes.

Clearly, a modi rate of zero (0.0) corresponds to an MPT program tree that will only produce a single output from the root to a particular class, which is very much the same as the standard program tree. A modi rate of one (1.0) would result in an MPT tree with many modi nodes, representing a large DAG during evaluation.

3.5 Genetic Operators

Although the MPT programs can represent powerful DAGs, which are suitable for multiclass object classification, the MPT programs still retain the main properties of the standard program trees in GP. Accordingly, with the new program structure, the genetic operators such as crossover, mutation, selection or reproduction will remain the same as in the standard GP system during the evolutionary process, making the implementation reasonably simple. In other words, the modi nodes, just like other normal nodes, can be mutated during evolution.

3.6 Main Characteristics

The new program structure has a number of nice properties. The new structure can represent a complex DAG and directly produce multiple related outputs. The classifiers of this kind can determine the class of an input object by simple voting of the multiple output values just as in neural networks. In this way, the complex translation from a single floating point value to multiple class labels is successfully avoided. At the same time, during evolution, an MPT program with the new program structure is just like a standard program tree, and thus has the advantages of the standard GP systems, for example, easy selection and recombination during the evolutionary process as well as easy implementation.

Examining the DAG representation of the MPT programs in Fig. 3 shows that the MPT programs can also have multilayers in a DAG, where leaf nodes form the primary features from the input objects, internal nodes extract higher level features and output nodes are associated with class labels. This is very similar to feed-forward neural networks. However, the DAG representation of the MPT programs differs from the feedforward neural networks in several aspects. First, the MPT DAGs allow unbalanced structure and non-full connections between neighbouring layers, which is much more flexible. Second, the internal (hidden) nodes and layers in the MPT DAGs are automatically evolved during the evolutionary process according to a particular object classification task, rather than being predefined based on *a priori* knowledge from the task domain or an empirical search through experiments as usual in feedforward neural networks. In addition, the MPT DAG structure also allows the *reuse* of child nodes. Every left most child of a modi node is reused by the modi node itself and the parent of the modi node, resulting in a two-way reuse. Multiway reuse is also possible by a sequence of hierarchically connected modi nodes, as shown in the bottom left corner around the feature terminal node *W*. Finally, from the GP point of view, the subtrees at the modi nodes play a role of automatically defined functions (ADFs), which are evolved together with the program trees [5].

4 Experiment Design and Configuration

4.1 Image Data Sets

In the experiment, we used four data sets providing object classification problems of varying difficulty. Example images are shown in Fig. 4.

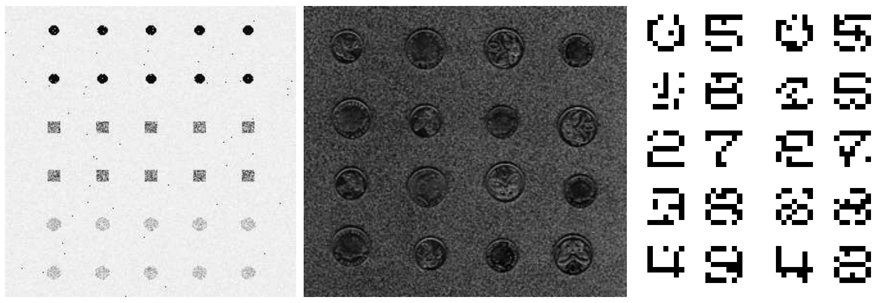


Fig. 4 Example data sets. (a) Shapes; (b) Coins; (c) Digits15; (d) Digits30

The first set of images (Fig. 4a) was generated to give well-defined objects against a relatively clean background. The pixels of the objects were produced using a Gaussian generator with different means and variances for each class. Three classes of 960 small objects were cut from those images to form the classification data set. The three classes are black circles, grey squares and light circles. For presentation convenience, this data set is referred to as *shape*.

The second set of images (Fig. 4b) contains scanned 5 cent and 10 cent New Zealand coins. The coins were located in different places with different orientations and appeared in different sides (heads and tails). In addition, the background was cluttered. We need to distinguish different coins with different sides from the background. Five classes of 801 object cutouts were created: 160 five-cent heads, 160 five-cent tails, 160 ten-cent heads, 160 ten-cent tails and the cluttered background (161 cutouts). Compared with the *shape* data set, the classification problem in this data set is much harder. Although these are still regular, man-made objects, the problem is very hard due to the low resolution and the noisy background with similar brightness to the coin objects in the images.

The third and fourth data sets are two-digit recognition tasks, each consisting of 1000 digit examples. Each digit example is an image of a 7×7 bitmap. In these two-digit recognition tasks, the goal is to automatically recognize which of the 10 classes (digits 0, 1, 2, ..., 9) each pattern (digit example) belongs to. Note that all the digit patterns have been corrupted by noise. In the two tasks (Figs. 4c and 4d), 15% and 30% of pixels, chosen at random, have been flipped. In data set 3, while some patterns can be clearly recognized by human eyes such as “0”, “2”, “5”, “7” and possibly “4”, it is not easy to distinguish between “6”, “8”, “3”, and even “9”, or even “1” and “5”. The task in data set 4 is even more difficult—human eyes cannot recognize a majority of the patterns, particularly “8”, “9” and “3”, “5” and “6”, and even “1”, “2” and “0”. In addition, the number of classes is much larger than those in tasks 1 and 2, making the two tasks even more difficult.

For all four data sets, the objects were equally split into three separate data sets: one-third for the training set used directly for learning the genetic program classifiers, one-third for the validation set for controlling overfitting and one-third for the test set for measuring the performance of the learned program classifiers.

4.2 Terminals, Functions and Fitness Functions in GP systems

In this approach, the feature terminals consisted of four local statistical features extracted from the object cutout examples in the first two tasks, and just 49 pixel values in the third and fourth tasks. While these features are very simple at quite a low level and definitely not the best ones for these tasks, they could perform reasonably well. Finding and extracting the best features for them is beyond the goal of this chapter. In addition, we also used constants in the terminal set, similar to the common setting.

The function set consisted of four standard arithmetic operators and a conditional operator. The division operator represents a “protected” division in which division by zero gives a result of zero. The conditional operator returns its second argument if its first argument is positive, and otherwise returns its third argument. While these operators are not the best for all these tasks, they are commonly used as low-level functions for classification. As it is not our goal to find specific functions towards the best performance for a particular task, we used this common setting.

In the object classification tasks used here, different classes have relatively evenly distributed object instances. Accordingly, we used the classification accuracy of a task on the training set as the fitness measure. For the new approach with the new program structure, the evaluation and classification process of an MPT program for a given object instance is described in Sect. 3.2. For the standard GP approach, for baseline comparison purpose, the commonly used program classification map or static range selection method [25, 26] was used to determine which class an input object instance belongs to based on the single output of the evolved program, as described in Sect. 2.3.

4.3 GP Parameters and System Configuration

For both the new approach (GP with the new program structure) described in this chapter and the standard (basic) GP approach, the ramped half-and-half method [6] was used to generate the initial population and for the mutation operator. For the new approach, a modi rate was also used as described in Sect. 3.4. The proportional selection mechanism and the reproduction [19], crossover and mutation operators [6] were used in the evolutionary learning process for both approaches. We used reproduction, mutation and crossover rates of 10%, 30% and 60%, respectively. The program depth was initialized from 3 to 5, and can be increased to 7 during evolution. The population size was 500. The evolutionary process was run for a maximum of 50 generations, unless it found a program that solved the problem perfectly (100% accuracy), at which point the evolution was terminated early.

Each of the experiments on all data sets for both approaches was repeated for 50 runs and the average results are presented (the standard deviations were quite small, and so we omitted them in the results tables and figures to give a clear view).

5 Results and Discussion

To investigate the performance of the new program structure, the new approach is examined and compared with the basic GP approach on the four data sets under the same experimental setting described in the previous section. This section first describes the best object classification results achieved by the two approaches, then presents the effect of the modi rate parameter in the new approach, followed by further discussions on some related issues.

5.1 Object Classification Performance

The average results of the best programs evolved by both approaches on the four data sets as well as the improvement of the new approach over the basic GP approach are shown in Table 1.

As can be seen from Table 1, both approaches did well for the shape data set as the task is relatively easy. In particular, both approaches almost achieved perfect results. For the coin data set, the new approach achieved 93.89% accuracy, 8.67% higher than the basic GP approach. For the two-digit data sets, the new approach

Table 1 Results of the new GP approach over the basic GP approach

Methods/ improvement	Data sets			
	Shape	Coin	Digit15	Digit30
Basic GP approach (%)	99.40	85.22	56.85	44.09
New GP approach (%)	99.77	93.89	68.11	54.46
Improvement (%)	0.37	8.67	11.26	10.37

performed much better than the basic GP, with improvements of more than 10%. In particular, for task 4, where even human eyes could only recognize a small proportion of the digit examples, the GP approach with MPT program structure could recognize a majority of them, achieving 54.46% accuracy. These results confirm that the GP approach with the new program structure performs better than the basic GP approach for these object classification programs, particularly for the relatively difficult tasks. These results suggest that the new program structure is more effective for multiclass object classification problems than the standard program structure in tree-based GP.

5.2 *The Effect of Modi Rate μ*

To investigate the effect of modi rates on the performance of the new program structure, we performed experiments on the four data sets using different modi rates ranging from 0.0 to 1.0. The results are shown in Fig. 5 in terms of the improvement (increase in classification accuracy) of the new GP approach over the basic GP approach.

According to Fig. 5, particularly for the three relatively difficult tasks, the modi rate does affect the system performance—the use of different modi rates leads to different classification performance. The influence of modi rates is not consistently

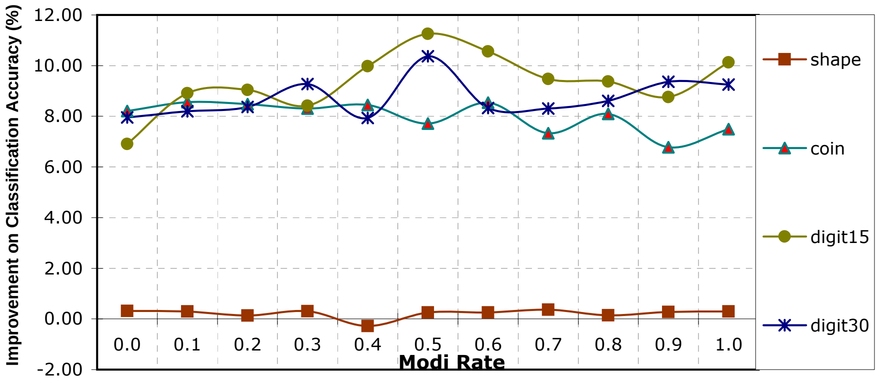


Fig. 5 Effect of different modi rates

proportional across different tasks. However, the results suggest that neither too big nor too small modi rates are good. It does not seem to have a reliable way of choosing a very appropriate modi rate for a task, and therefore empirical search through experiments is usually needed. However, if such a search can improve performance significantly, this will be a small price to pay. The experiments suggest that for the three relatively difficult tasks, a modi rate of 0.5 (or in the range 0.3–0.6) can serve as a good starting point.[1]

The "middle-range modi rates are good" observation suggests that for initializing the population, programs with not too many modi nodes or not too few modi nodes are convenient for GP to *start evolution*. This is reasonable because one interpretation of the modi rate is the expected proportion of *reuses* and the expected number of updates of the output vector in the initial programs. Too low reuse and/or too few updates of the output vector will definitely reduce the power of the program, whereas too high a rate will make the reuse and update effect unnecessarily messy.

We originally hypothesized a small modi rate such as 0.0 or 0.1 would lead to very bad results as zero modi rate would mean that there is only one modi node in the MPTs connected to a single class before evolution. However, the effect of modi rates (see Fig. 5) was clearly not as large as we originally thought. This is probably due to the use of a large size of population (500): the population as a whole provides "sufficient" modi nodes that the *best* evolved program can use through genetic operators such as crossover. In addition, the mutation operator might also generate modi nodes during evolution.

5.3 Discussion on Program Size

The results of the two-digit data sets are worse than the results of the shape and the coin data sets. This is mainly due to the difficulty of the classification problems and nature of these problems. In these problems, there are not only considerably more features but also a much larger number of classes, compared with the first two object classification problems. On the other hand, we used a relatively small program size (maximum program tree depth of 7) for the two-digit data sets. Compared with the first two object classification problems, the evolved programs would have many fewer chances to include each of the 49 features as leaf nodes for handling input features. In addition, as there are a large number of classes (10 in the two problems), the probability of producing sufficient modi nodes for different classes would be much smaller than the first two object classification tasks. However, in both GP approaches, a program depth of 7 should permit us to have more than twice as many nodes as the input features. It is not clear whether an increase in program size can lead to better performance—this needs to be further investigated in the future.

[1] Experiments on another ten data sets with multiclass object classification tasks also show a similar pattern. Due to space constraints, we do not present them here.

6 Conclusions

This chapter described a new way of using the tree-based genetic program structure in GP for multiclass object classification. This was done through the new program structure MPT, which uses a *virtual* output vector to produce multiple values, each of which corresponds to a class, instead of using the root node of a tree to produce a single number. The complex translation of the single number into different regions for multiclass was successfully avoided. This approach was examined and compared with the basic GP approach on four object classification problems of increasing difficulty. Results showed that the new approach outperformed the basic approach on all these tasks.

The results also showed that different modi rates led to different results. Neither too small nor too large modi rates were good. However, there did not seem to exist an efficient and reliable way of choosing a good modi rate for a particular problem. Rates between 0.3 and 0.6 could serve as a good starting point.

Although developed for object classification problems, this approach is expected to be general and to be applied to other classification problems with three or more classes.

For future work, we will investigate the new approach for the digit recognition tasks with a larger program size and examine whether the performance can be improved. We will also examine this approach on other multiclass classification problems and compare this approach with other long established learning techniques such as decision trees and neural networks.

Acknowledgements. We thank Yun Zhang for some ideas and experiments when she was working as a project student and a research assistant. We would also like to thank Malcolm Lett and Will Smart in our GP group and members in the AI research group, particularly Peter Andreae and Marcus Frean, for a number of useful discussions.

This work was supported in part by the Marsden Fund council from the government funding (05-VUW-017), administrated by the Royal Society of New Zealand.

References

1. Holland, J.H. (ed.): Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence. University of Michigan Press/ MIT Press, Ann Arbor, Cambridge (1975)
2. Michalewicz, Z.: Genetic algorithms + data structures = evolution programs, 3rd edn. Springer, London (1996)
3. Friedberg, R.: A learning machine, Part I. IBM Journal of Research and Development 2, 2–13 (1958)
4. Koza, J.R.: Genetic programming: on the programming of computers by means of natural selection. MIT Press, Cambridge (1992)
5. Koza, J.R.: Genetic Programming II: Automatic Discovery of Reusable Programs. MIT Press, Cambridge (1994)

6. Banzhaf, W., Nordin, P., Keller, R.E., Francone, F.D.: Genetic Programming: An Introduction on the Automatic Evolution of computer programs and its Applications. In: Subject: Genetic programming (Computer science). Morgan Kaufmann Publishers, San Francisco (1998)
7. MacKay, D.J.C.: Information Theory, Inference, and Learning Algorithms. Cambridge University Press, Cambridge (2003)
8. Banzhaf, W.: Genetic programming for pedestrians. In: Forrest, S. (ed.) Proceedings of the 5th International Conference on Genetic Algorithms, ICGA 1993, University of Illinois at Urbana-Champaign, July 17-21, p. 628. Morgan Kaufmann, San Francisco (1993)
9. Jacob, C.: Evolving evolution programs: Genetic programming and L-systems. In: Koza, J.R., Goldberg, D.E., Fogel, D.B., Riolo, R.L. (eds.) Genetic Programming 1996: Proceedings of the First Annual Conference, Stanford University, CA, USA, July 28–31, pp. 107–115. MIT Press, Cambridge (1996)
10. Teller, A., Veloso, M.: PADO: Learning tree structured algorithms for orchestration into an object recognition system. Technical Report CMU-CS-95-101, Department of Computer Science, Carnegie Mellon University, Pittsburgh, PA, USA (1995)
11. Kantschik, W., Banzhaf, W.: Linear-graph GP—A new GP structure. In: Foster, J.A., Lutton, E., Miller, J., Ryan, C., Tettamanzi, A.G.B. (eds.) EuroGP 2002. LNCS, vol. 2278, pp. 83–92. Springer, Heidelberg (2002)
12. Whigham, P.A.: Grammatically-based genetic programming. In: Rosca, J.P. (ed.) Proceedings of the Workshop on Genetic Programming: From Theory to Real-World Applications, Tahoe City, California, USA, July 9, pp. 33–41 (1995)
13. Nordin, P.: A compiling genetic programming system that directly manipulates the machine code. In: Kinnear Jr., K.E. (ed.) Advances in Genetic Programming, ch. 14, pp. 311–331. MIT Press, Cambridge (1994)
14. Eggermont, J., Eiben, A.E., van Hemert, J.I.: A comparison of genetic programming variants for data classification. In: Hand, D.J., Kok, J.N., Berthold, M.R. (eds.) IDA 1999. LNCS, vol. 1642, p. 281. Springer, Heidelberg (1999)
15. Howard, D., Roberts, S.C., Ryan, C.: The boru data crawler for object detection tasks in machine vision. In: Cagnoni, S., Gottlieb, J., Hart, E., Middendorf, M., Raidl, G.R. (eds.) EvoIASP 2002, EvoWorkshops 2002, EvoSTIM 2002, EvoCOP 2002, and EvoPlan 2002. LNCS, vol. 2279, pp. 220–230. Springer, Heidelberg (2002)
16. Song, A., Ciesielski, V., Williams, H.: Texture classifiers generated by genetic programming. In: Fogel, D.B., El-Sharkawi, M.A., Yao, X., Greenwood, G., Iba, H., Marrow, P., Shackleton, M. (eds.) Proceedings of the 2002 Congress on Evolutionary Computation CEC 2002, pp. 243–248. IEEE Press, Los Alamitos (2002)
17. Tackett, W.A.: Genetic programming for feature discovery and image discrimination. In: Forrest, S. (ed.) Proceedings of the 5th International Conference on Genetic Algorithms, ICGA 1993, University of Illinois at Urbana-Champaign, July 17-21, pp. 303–309. Morgan Kaufmann, San Francisco (1993)
18. Zhang, M., Andreae, P., Pritchard, M.: Pixel statistics and false alarm area in genetic programming for object detection. In: Raidl, G.R., Cagnoni, S., Cardalda, J.J.R., Corne, D.W., Gottlieb, J., Guillot, A., Hart, E., Johnson, C.G., Marchiori, E., Meyer, J.-A., Middendorf, M. (eds.) EvoIASP 2003, EvoWorkshops 2003, EvoSTIM 2003, EvoROB/EvoRobot 2003, EvoCOP 2003, EvoBIO 2003, and EvoMUSART 2003. LNCS, vol. 2611, pp. 455–466. Springer, Heidelberg (2003)

19. Zhang, M., Ciesielski, V., Andreae, P.: A domain independent window-approach to multiclass object detection using genetic programming. EURASIP Journal on Signal Processing, Special Issue on Genetic and Evolutionary Computation for Signal Processing and Image Analysis 2003(8), 841–859 (2003)
20. Zhang, M., Smart, W.: Using gaussian distribution to construct fitness functions in genetic programming for multiclass object classification. Pattern Recognition Letters 27(11), 1266–1274 (2006)
21. Lin, Y., Bhanu, B.: Object detection via feature synthesis using MDL-based genetic programming. IEEE Transactions on Systems, Man and Cybernetics, Part B 35(3), 538–547 (2005)
22. Song, A., Loveard, T., Ciesielski, V.: Towards genetic programming for texture classification. In: Stumptner, M., Corbett, D.R., Brooks, M. (eds.) Canadian AI 2001. LNCS, vol. 2256, pp. 461–472. Springer, Heidelberg (2001)
23. Tackett, W.A.: Recombination, Selection, and the Genetic Construction of Computer Programs. PhD thesis, Faculty of the Graduate School, University of Southern California, Canoga Park, California, USA (April 1994)
24. Zhang, M., Ciesielski, V.: Genetic programming for multiple class object detection. In: Foo, N.Y. (ed.) Canadian AI 1999. LNCS (LNAI), vol. 1747, pp. 180–192. Springer, Heidelberg (1999)
25. Loveard, T., Ciesielski, V.: Representing classification problems in genetic programming. In: Proceedings of the Congress on Evolutionary Computation, COEX, World Trade Center, 159 Samseong-dong, Gangnam-gu, Seoul, Korea, May 27-30, vol. 2, pp. 1070–1077. IEEE Press, Los Alamitos (2001)
26. Zhang, M., Smart, W.: Multiclass object classification using genetic programming. In: Raidl, G.R., Cagnoni, S., Branke, J., Corne, D.W., Drechsler, R., Jin, Y., Johnson, C.G., Machado, P., Marchiori, E., Rothlauf, F., Smith, G.D., Squillero, G. (eds.) EvoWorkshops 2004. LNCS, vol. 3005, pp. 369–378. Springer, Heidelberg (2004)
27. Song, A.: Texture Classification: A Genetic Programming Approach. PhD thesis, Department of Computer Science, RMIT University, Melbourne, Australia (2003)
28. Roberts, M., Claridge, E.: A multi-stage approach to cooperatively coevolving image feature construction and object detection. In: Rothlauf, F.R., et al. (eds.) Applications of Evolutionary Computation, Proceedings of Evolutionary Computing in Image and Signal Analysis (EvoIASP), Lausanne Switzerland, March 30- April 1. LNCS, vol. 3003. Springer, Heidelberg (2005)
29. Montana, D.J.: Strongly typed genetic programming. Evolutionary Computation 3(2), 199–230 (1995)
30. Forsyth, D.A., Ponce, J.: Computer Vision: A Modern Approach. Prentice-Hall, Englewood Cliffs (2003)
31. Caelli, T., Bischof, W.F.: Machine Learning and Image Interpretation. Plenum Press, New York (1997)
32. Gose, E., Johnsonbaugh, R., Jost, S.: Pattern Recognition and Image Analysis. Prentice Hall PTR, Upper Saddle River (1996)
33. Howard, D., Roberts, S.C., Brankin, R.: Target detection in SAR imagery by genetic programming. Advances in Engineering Software 30, 303–311 (1999)
34. Teller, A., Veloso, M.: A controlled experiment: Evolution for learning difficult image classification. In: Pinto-Ferreira, C., Mamede, N.J. (eds.) EPIA 1995. LNCS (LNAI), vol. 990, pp. 165–176. Springer, Heidelberg (1995)

35. Robinson, G., McIlroy, P.: Exploring some commercial applications of genetic programming. In: Fogarty, T.C. (ed.) AISB-WS 1995. LNCS, vol. 993. Springer, Heidelberg (1995)
36. Isaka, S.: An empirical study of facial image feature extraction by genetic programming. In: Koza, J.R. (ed.) The Genetic Programming 1997 Conference, Stanford Bookstore, Stanford University, CA, USA, pp. 93–99 (July 1997) (Late Breaking Papers)
37. Andre, D.: Automatically defined features: The simultaneous evolution of 2-dimensional feature detectors and an algorithm for using them. In: Kinnear, K.E. (ed.) Advances in Genetic Programming, pp. 477–494. MIT Press, Cambridge (1994)
38. Ciesielski, V., Innes, A., John, S., Mamutil, J.: Understanding evolved genetic programs for a real world object detection problem. In: Keijzer, M., Tettamanzi, A.G.B., Collet, P., van Hemert, J., Tomassini, M. (eds.) EuroGP 2005. LNCS, vol. 3447, pp. 351–360. Springer, Heidelberg (2005)
39. Song, A., Ciesielski, V.: Texture analysis by genetic programming. In: Proceedings of the 2004 IEEE Congress on Evolutionary Computation, Portland, Oregon, June 20-23, pp. 2092–2099. IEEE Press, Los Alamitos (2004)
40. Bhanu, B., Lin, Y.: Object detection in multi-modal images using genetic programming. Applied Soft Computing 4(2), 175–201 (2004)
41. Bhanu, B., Lin, Y., Krawiec, K.: Evolutionary Synthesis of Pattern Recognition Systems. In: Monographs in Computer Science. Springer, New York (2005)
42. Cagnoni, S., Lutton, E., Olague, G.: Genetic and Evolutionary Computation for Image Processing and Analysis. In: EURASIP Book Series on Signal Processing and Communications, vol. 8. Hindawi Publishing Corporation (to appear) (2007)
43. Olaguea, G., Cagnoni, S., Lutton, E. (eds.): special issue on evolutionary computer vision and image understanding. Pattern Recognition Letters 27(11) (2006)
44. Rumelhart, D.E., Hinton, G.E., Williams, R.J.: Learning internal representations by error propagation. In: Rumelhart, D.E., McClelland, J.L., the PDP research group (eds.) Parallel distributed Processing, Explorations in the Microstructure of Cognition. Foundations, vol. 1, ch. 8. MIT Press, Cambridge (1986)

Genetic Programming for Generative Learning and Recognition of Hand-Drawn Shapes

Wojciech Jaśkowski, Krzysztof Krawiec, and Bartosz Wieloch

Abstract. We propose a novel method of evolutionary visual learning that uses a generative approach to assess the learner's ability to recognize image contents. Each learner, implemented as a genetic programming (GP) individual, processes visual primitives that represent local salient features derived from the input image. The learner analyzes the visual primitives, which involves mostly their grouping and selection, eventually producing a hierarchy of visual primitives build upon the input image. Based on that it provides partial reproduction of the shapes of the analyzed objects and is evaluated according to the quality of that reproduction. We present the method in detail and verify it experimentally on the real-world task of recognition of hand-drawn shapes. In particular, we show how GP individuals trained on examples from different decision classes can be combined to build a complete multiclass recognition system. We compare such recognition systems to reference methods, showing that our generative learning approach provides similar results. This chapter also contains detailed analysis of processing carried out by an exemplary individual.

1 Introduction

In supervised learning applied to object recognition, the search in the space of hypotheses (learners) is usually guided by some measure of quality of discrimination of training examples from different object classes. This requires defining, somehow arbitrarily, learner's desired response for objects from particular classes, e.g., by specifying the desired combinations of output layer excitations in case of an artificial neural network. Though proven effective in several applications, such an approach suffers from relatively high risk of overfitting, especially when the number of image features is large. For instance, in past experience with evolutionary synthesis of object recognition systems [1, 2], many evolved learners tended to use irrelevant

Wojciech Jaśkowski, Krzysztof Krawiec, and Bartosz Wieloch
Institute of Computing Science, Poznan University of Technology,
Piotrowo 2, 60965 Poznań, Poland
e-mail: {wjaskowski,kkrawiec,bwieloch}@cs.put.poznan.pl

image features, coincidentally correlated with the partitioning of training examples into concepts. This happens because learners are rewarded exclusively for decisions they make, and not for the actual ‘understanding’ of the recognized objects.

Moreover, applicability of such supervised feature-based learning methods is restricted to simple recognition tasks with a limited number of object classes. For more complex objects and for large numbers of object classes, one usually has to rely on model-based approach and explicitly specify the models of objects to be recognized, which is often tedious and time consuming.

To avoid overfitting, on one hand, and the model-based approach, on the other hand, in this chapter, we make the learning process unsupervised in the sense that the learner is not told arbitrarily what kind of output it should produce in response to particular training examples. Rather, it is encouraged to reproduce the shape of the object being recognized, which, in turn, enables a more thorough evaluation. To implement such learners, we use genetic programming (GP), [3, 4]. GP is a variant of evolutionary computation (EC), a general bio-inspired template for performing global parallel search in high-dimensional spaces [5, 6]. Its commonly quoted virtues include relatively little task-specific tailoring and low risk of being trapped in local minima of objective function. It has sound rationale in both computational biology and in optimization theory, and has proven effective in a wide spectrum of benchmarks and real-world applications [7]. We use it here mostly because our space of genetic programs (i.e. the space of all possible learners-hypotheses) cannot be effectively searched by means of exact methods due to high dimensionality and enormous cardinality. Also, our objective function lacks properties that could be used to speed up the search by, for instance, pruning the unpromising parts of the search space (as in the Branch and Bound algorithm). Heuristic or metaheuristic search is, therefore, the only plausible method that can yield reasonably good suboptimal solutions in a polynomial time. This is also consistent with the practical attitude chosen here, where we do not strain to necessarily find a globally optimal recognizer, being satisfied with a well-performing suboptimal one.

The primary contribution of this chapter is thus a novel method to object recognition that (i) uses GP to evolve visual learners, (ii) estimates learner’s fitness by assessing its ability to restore the essential features of the input image and (iii) uses visual primitives as basic ‘granules’ of visual information. The chapter details this approach and is organized as follows. In Sect. 2, we outline the related research on evolutionary visual learning. Section 3 presents motivations for our design of the visual learner. In Sect. 4, we detail the approach, and in Sect. 5, we apply it to the real-world task of handwritten shape recognition. Section 6 concludes the chapter.

2 Related Research in Visual Learning

In most approaches to visual learning reported in literature, learning is limited to parameter optimization and usually concerns only a particular step of information processing, such as image preprocessing, segmentation or feature extraction. Learning methods that use raw image data and produce complete recognition systems are rare, often make use of domain-specific knowledge and/or predefined object models, and are usually specialized towards a particular application. Only a few methods close the feedback loop of the learning process at the outermost (e.g. recognition) level [2, 8–14]. In the following, we review the selected past contributions that belong to this category and use GP as the underlying learning paradigm.

Tackett [8] applied GP to detection of vehicles in IR imagery and compared the evolved solutions to artificial neural networks and classification tree. GP performed the best and offered much faster execution times, though at the expense of increased learning effort. Johnson *et al*. [9] used a variant of GP for locating hands in images representing human body silhouettes. The evolved program was able to correctly classify up to 93% of the images, which was better than the best algorithm the authors were able to write by hand. Teller and Veloso [15] proposed a GP variant with individuals encoded by graphs (as opposed to trees in canonical GP) and applied it successfully to face recognition in greyscale images. Poli [16] applied GP to discover efficient optimal filters that solve problems of image enhancement, feature detection and image segmentation. The method was used to segment brain images and compared with artificial neural nets. Daida *et al.* [10] applied GP with sophisticated dynamic fitness to detect specific features in SAR images of the Arctic Ocean. Winkeler and Manjunath [11] examined GP as a machine learning technique for face detection, interestingly combining two programs evolved in separate experiments that used different features to boost the detection rate. The resulting detector significantly improved the results when compared to single feature detectors.

In [13], Howard et al. investigated automatic detection of ships in low-resolution synthetic aperture radar (SAR) imagery. The objectives were to maximize detection accuracy across multiple images, to minimize the computational effort during image processing and to minimize the effort during the design stage. The results demonstrated that GP successfully evolves detectors that satisfy these objectives. In [17], Howard and Roberts proposed sophisticated multi-stage GP terminals based on Fourier transforms. The extended approach turned out to be better when applied to the task of detection of parked vehicles in aerial imagery. Rizky, et al. [12] applied hybrid evolutionary computation (GP combined with neural networks) to evolve recognition systems (mostly feature detectors) and approached the demanding real-world task of object recognition in radar modality based on one-dimensional signals called radar signatures (a few thousands of views of six airborne objects). The approach evolved three components of the recognition system: properties (parameters) of feature detectors, structural form of the transformation and selection of feature detectors. The obtained results were encouraging when compared to baseline approaches (nearest neighbour, neural network and radial basis functions).

It is widely accepted in the pattern recognition community that, for many applications, the accuracy of classification (recognition ratio) as a performance measure of the recognition system is not sophisticated enough, as it naively combines both true positives and true negatives. Inspired by this observation, Lett and Zhang [18] proposed two new fitness functions based on recall and precision and used them to evolve GP procedures. In a similar spirit, Krawiec [19] developed a multiobjective GP-based approach, measuring recognizer's performance separately on each training example. This evaluation scheme provided a more thorough search and produced better-performing recognition systems when applied to the task of recognition of handwritten characters.

In [1, 2], we proposed a method that evolved feature extraction procedures encoded as GP and linear GP individuals. The approach implemented feature-based recognition, with each individual in population encoding a particular sequence of predefined image processing and feature extraction operations. When undergoing evaluation, an individual produced feature values for all images from the training set. Next, the discriminative ability of the resulting features was measured by performing an internal cross-validation test on the training data using a machine learning classifier.

The idea of symbolic processing of attributed visual primitives (VPs) using GP, which is the core element of the approach presented in this chapter, was first explored in [20], where we applied it to recognize computer screens in indoor scenes. Despite encouraging results, it was obvious that solving such a specific task cannot lead to elaboration of more general visual concepts and generic understanding of images. This shortcoming motivated work presented in this chapter.

The approach presented here may be considered as a special case of *generative* pattern recognition. In a representative study on that topic, Revow *et al.* used a predefined set of deformable models encoded as B-splines and an elastic matching algorithm based on expectation maximization to recognize handwritten characters [21]. In [22], an analogous approach proved useful for recognizing hand-drawn shapes. However, our method goes further, as it does not use explicit models of objects but encodes them implicitly in the GP code. Second, our 'implicit models' are not handcrafted but automatically derived from the training data in the process of evolutionary learning. In addition, last but not least, our learners have to reproduce the input image using *multiple* drawing actions, which forces them to discover how to break up the analyzed shapes into elementary components.

3 Motivations for Generative Learning

Machine learning algorithms learn by performing a search in the space of hypotheses (identified here with *learners* and *individuals*) [23]. This process requires some guidance so that the finally selected hypothesis performs well on both training and testing set. In supervised learning, such a guidance is usually provided by quality of discrimination of decision classes, technically implemented as classification accuracy, sensitivity, selectivity or a similar measure. This approach is characteristic for, among others, the 'wrapper' approach to feature selection and construction in machine learning [24].

The working supposition of this chapter is that such a way of evaluating learner's performance is superficial, as it focuses exclusively on learner's *final* output and does not examine the processing that led to it. If the number of possible hypotheses is high and the number of training examples is low, which is often the case in evolutionary visual learning that usually involves time consuming fitness function, many hypotheses may perform well by pure chance despite, for instance, relying on object features that are essentially irrelevant for the task being solved. This, in turn, leads to low recognition performance on the test set.

If, in addition to feature selection, learning also involves feature *synthesis*, i.e. explicit construction of a mapping from the space of raster images into the space of image features, overfitting becomes even more likely. For instance, in our past experience with evolutionary design of pattern recognition systems [1, 2], a recognition system might evolve an irrelevant feature that was coincidentally correlated with the decision class label. Overfitting is also likely to lead to false positive errors when one classifies objects beyond the problem domain, i.e. examples that do not belong to any of decision classes that the system was trained on. For this reason, Revow *et al.*, when dealing with handwritten character recognition, prefer generative methods to statistical ones, stating that "statistical recognizers can occasionally confidently classify images that do not look like a character" [21, p. 593].

Putting the above argument in another perspective, learner's evaluation in terms of class discrimination only does not necessarily reflect the 'appropriateness' of its entire decision-making process, particularly of the features being used. To circumvent this problem, in our approach, learner is not expected to explicitly discriminate the positive examples from the negative ones. Rather, it should *learn the generative description of the positive class*. To attain that, it has to detect the important image features and based on them, produce a sketch that reproduces the input shape. The reproduction process is sequential and consists of a series of *drawing actions* issued by a GP individual, which reproduce the input shape part by part. In such a way, learner undergoes a more thorough evaluation when compared to conventional supervised learning, where only final decisions are considered.

It should be noted that such procedural shape reproduction reflects to some extent the process of human image understanding, especially if the subject of reasoning is *shape*. For instance, a person asked to prove his/her understanding of the concept of a triangle is expected to formulate it as a specific arrangement of simpler visual objects—sections. When confronted with an example of a triangle, such a person is not only able to recognize it, but also to produce a top-down decomposition, by identifying the particular components (three sections) and showing how the triangle can be reproduced from them. This clearly indicates the procedural character of the underlying recognition process. Similarly, our method is procedural and generative: the learning agent is expected to *generate* drawing in response to the input stimulus.

4 Generative Visual Learning

The central and novel feature of the approach described in this chapter is that our evolving recognizers learn by trying to reproduce the input image and are rewarded according to the quality of that reproduction. This feature allows us to term the approach *generative visual learning*. Technically, reproduction takes place on a virtual canvas spanned over the input image. On that canvas, the learner is allowed to perform some elementary *drawing actions* (DAs). In this chapter, DAs boil down to sections, which are sufficient to perform successful reproduction of hand-drawn polygons considered in the experimental part. It should be therefore emphasized in the beginning that the reproduction process is partial, i.e. learners restore only a particular *aspect* of image contents. In this chapter, the aspect of interest is shape, whereas other features, related to colour or texture, are ignored.

As an illustration, let us consider the process of reconstruction of an empty triangular shape. It requires the learner to perform the following steps: (i) detection of conspicuous features—triangle corners— (ii) pairing of the detected triangle corners and (iii) performing DAs that connect the paired corners. However, within the proposed approach, the learner is not given *a priori* information either about the concept of the corner or about the expected number of them. We expect the learner to discover these.

The method abstracts from raster data and relies only on selected salient features in the input image s. This helps to reduce the amount of data that has to be processed and to bias the learning towards the image aspect of interest (shape). For each locally detected feature, we build an independent VP. The complete set of VPs derived from s is denoted hereafter by P. The learning algorithm itself does not make any assumptions about the particular salient feature used for VP creation. Reasonable instances of VPs include, but are not limited to, edge fragments, regions, texems or blobs. However, the type of detected feature determines the image aspect that is reconstructed. As in this chapter we focus on shape, and use VPs representing prominent local luminance gradients derived from s using a straightforward procedure. Each VP is described by three scalars called *attributes* hereafter; these include two spatial coordinates of the edge fragment and the local gradient orientation.

The method uses an evolutionary algorithm [5, 6] to maintain a population of individuals (solutions), each of them being a visual learner implemented as a GP tree[3]. The processing carried out by a learner L for an input image s (stimulus) may be split into three stages (see Fig. 1). First, we detect the VPs from s and gather them in the set P. Next, L analyzes P and produces a drawing on a canvas c, attempting to reproduce the shape of object shown in s. Finally, canvas c is compared to the original input image s, and the result of that comparison determines the fitness of the individual-learner L. The better the canvas c matches the input image s, the higher the fitness is assigned to L.

Each learner L is composed of nodes representing *elementary functions* that process sets of VPs and terminal nodes (named *ImageNode*s) that fetch the set of primitives P derived from the input image s, and the consecutive internal nodes process the primitives, all the way up to the root node. Table 1 presents the complete

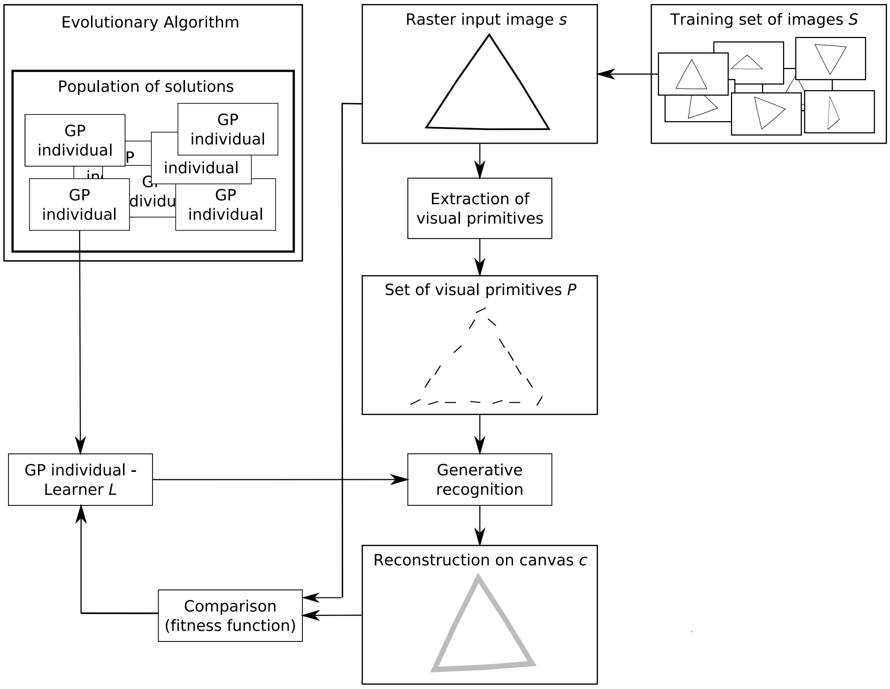

Evolutionary Algorithm
Population of solutions
GP individual
GP individual
individual
GP individual
GP individual
Raster input image s
Training set of images S
Extraction of visual primitives
Set of visual primitives P
GP individual - Learner L
Generative recognition
Reconstruction on canvas c
Comparison (fitness function)

Fig. 1 The outline of the approach

list of GP functions and terminals that may reside in particular tree nodes. We use strongly typed GP (cf. [3]), which implies that two nodes may be connected to each other only if their input/output types match. The following types are used: numerical scalars ($\mathfrak{R}$ for short), sets of VPs (Ω, potentially nested), attribute labels (A), binary arithmetic relations (R) and aggregators (G).

The GP functions may be divided into the following categories (see [25, 26] for detailed description):

(1) Scalar functions (as in standard GP applied to symbolic regression; see [3]). Scalar functions accept arguments of type $\mathfrak{R}$ and return result of type $\mathfrak{R}$.

(2) Selectors. The role of a selector is to filter out some of the VPs it receives from its child node(s) according to some objectives or condition. Selectors accept at least one argument of type Ω and return result of type Ω. *Non-parametric selectors* expect two child nodes of type Ω and produce an output of type Ω. GP functions that implement basic set algebra, such as set union, intersection or difference, belong to this category. *Parametric selectors* expect three child nodes of types Ω, A and $\mathfrak{R}$, respectively, and produce output of type Ω. For instance, the function *LessThan* applied to child nodes (P, p_o, 0.3) filters all VPs from P for which the value of the attribute p_o (orientation) is less than 0.3.

(3) Iterators. The role of an iterator is to process the VPs it receives from one of its children one by one. For instance, the function *ForEach* iterates over all the

Table 1 GP terminals and function set

Type	Function
$\Re$	Ephemeral random constant
Ω	*ImageNode*—the VP representation P of the input image s
A	p_x, p_y, p_o and custom attributes added by *AddAttribute*
R	*Equals, Equals5Percent, Equals10Percent, Equals20Percent, LessThan, GreaterThan*
G	Sum, Mean, Product, Median, Min, Max, Range
$\Re$	+($\Re$,$\Re$), –($\Re$,$\Re$), *($\Re$,$\Re$), /($\Re$,$\Re$), sin($\Re$), cos($\Re$), abs($\Re$), sqrt($\Re$), sgn($\Re$), ln($\Re$)
Ω	*SetIntersection*(Ω,Ω), *SetUnion*(Ω,Ω), *SetMinus*(Ω,Ω), *SetMinusSym*(Ω,Ω), *SelectorMax*(Ω,A), *SelectorMin*(Ω,A), *SelectorCompare*(Ω,A,R,$\Re$), *CreatePair*(Ω,Ω), *CreatePairD*(Ω,Ω), *ForEach*(Ω,Ω), *ForEachCreatePair*(Ω,Ω,Ω), *ForEachCreatePairD*(Ω,Ω,Ω), *AddAttribute*(Ω,$\Re$), *AddAttributeForEach*(Ω,$\Re$), *GroupHierarchyCount*(Ω,$\Re$), *GroupHierarchyDistance*(Ω,$\Re$), *GroupProximity*(Ω,$\Re$), *GroupOrientationMulti*(Ω,$\Re$), *Ungroup*(Ω), *Draw*(Ω)

VPs from its left child and processes each of them using the GP code specified by its right child. The VPs resulting from all iterations are grouped into one VP and returned.

(4) Grouping functions. The role of those functions is to group primitives according to some objectives or conditions. For instance, *GroupHierarchyCount* uses agglomerative hierarchical clustering, where Euclidean distance of primitives serves as the distance metric.

(5) Attribute constructors. An attribute constructor defines and assigns a new attribute to the VP it processes. The definition of a new attribute, which must be based on the values of existing VP attributes, is given by the GP code contained in the right child subtree. To compute the value of a new attribute, attribute constructor passes the VP (function *AddAttribute*) or the sub-primitives of the VP (function *AddAtributeToEach*) through that subtree. Attribute constructors accept one argument of type Ω and another of type $\Re$, and return a result of type Ω.

Given the elementary functions, a learner L applied to an input image s gradually builds a hierarchy of VP sets derived from s. Each application of the selector, iterator or grouping function creates a new set of VPs that includes other elements of the hierarchy. In the end, the root node returns a nested VP hierarchy built atop of P, which reflects the processing performed by L for s. Some of the elements of the hierarchy may be tagged by new attributes created by attribute constructors.

Figure 2 illustrates the selected steps of a hypothetical learner's processing, i.e. of building a VP hierarchy in response to an exemplary input image. Figure 2(i) shows the fragment of learner's code—a path from one of its leaves to the root node. The remaining subfigures illustrate the outcomes of particular tree nodes on that path in the bottom-up order. Figure 2(ii) shows P, the input image representation returned by the *ImageNode* function; it contains four VPs, each of them depicted as a short segment marked by a letter. This set of primitives is then fetched by GP function 1, which groups primitives b and c into one subset named p_1, illustrated in Fig. 2(iii) as a dashed-line oval. In Fig. 2(iv), we demonstrate the final result of processing, which includes three groups of primitives. In particular, each of the shapes, denoted by p_1,

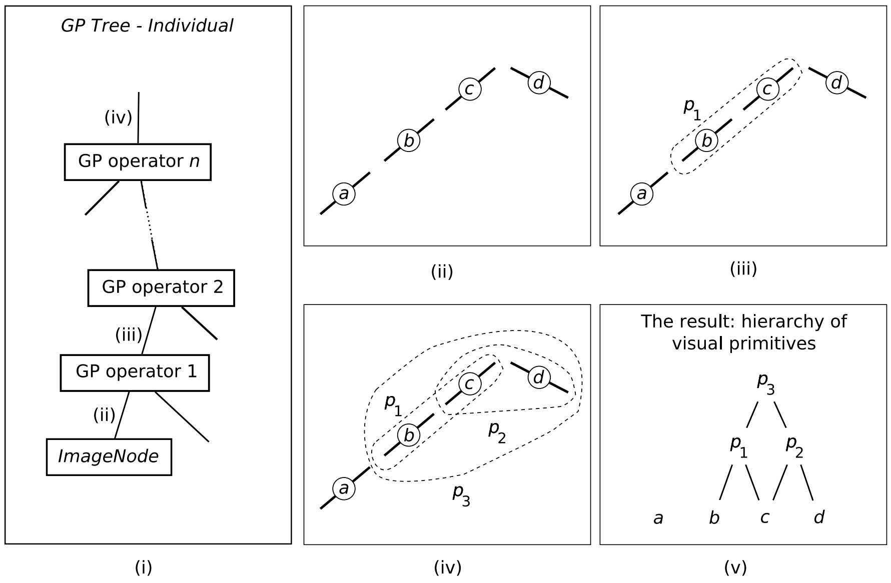


Fig. 2 The primitive hierarchy built by the learner from VPs, imposed on the image (*left*) and shown in an abstract form (*right*); VP attributes not shown for clarity

p_2, and p_3, corresponds to a single VP group built on this processing path. In Fig. 2(v), this resulting hierarchy is shown in an abstract way, without referring to the actual placement of particular VPs in the input image. Note that the hierarchy does not have to contain all VPs from P, and that a particular VP from P may occur more than once. Importantly, the primitive hierarchy (Fig. 2v) should not be confused with the GP tree (Fig. 2i).

An individual's fitness results from DAs that it performs in response to VP P derived from training images $s \in S$. To reconstruct the essential features of the input image s, the learner is allowed to perform DAs that boil down to drawing sections on the output canvas c. The simplest drawing action is implemented by the GP function called *Draw*. It expects as an argument one VP set T and returns it unchanged, drawing on canvas c sections connecting each pair of VPs contained in T. There are also a few other GP functions capable of performing DAs, not discussed here for brevity; they are included in the set of functions presented in Table 1 and marked by names ending with capital letter D.

The drawing created on the canvas c by a learner L for an input image s is evaluated by comparing c to s. For this purpose, we designed a simple and efficient algorithm based on a greedy heuristic. The algorithm assumes that the difference between c and s is proportional to the minimal total cost of bijective assignment of lit pixels of c to lit pixels of s. The total cost is defined as a sum of costs for each pixel assignment. The cost of pixel assignment depends on the distance between pixels: when the distance is less than 5, the cost is 0; maximum cost equals 1 when the distance is greater than 15; between 5 and 15, the cost is calculated as a linear

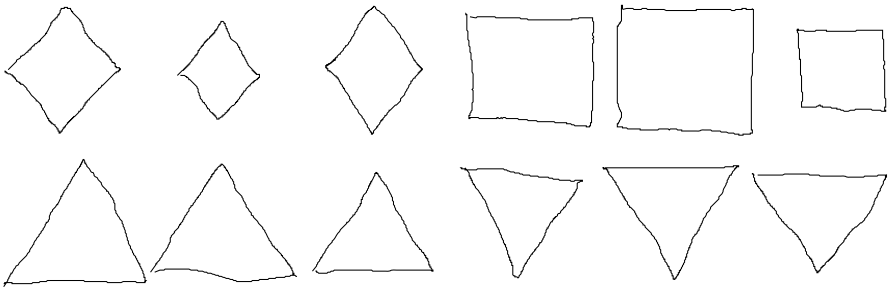


Fig. 3 Selected training examples

function of the distance. For pixels that cannot be assigned (e.g. because there are more lit pixels in c than in s), an additional penalty of value 1 is added to the total cost.

The (minimized) fitness of L is defined as the total cost of the assignment normalized by the number of lit pixels in $s \in S$, averaged over the entire training set of images S. The ideal learner perfectly restores shapes in all training images and its fitness amounts to 0. The more the canvas c produced by a learner differs from s, the greater its fitness value.

5 Experimental Evaluation

In the experimental part of this chapter, we tackle the problem of interpretation of hand-drawn sketches. This section describes our first attempts to apply this approach to real-world data, as opposed to [27], where its simplified and multi-objective variant has been trained on synthetic images of geometric figures. In real-world scenarios, automated interpretation of sketches may be helpful for direct digitization of diagrams, such as block diagrams and UML diagrams, saving the time required for tedious manual re-entry of paper notes. Such drawings are typically acquired using input devices such as TabletPC computers, PDAs or graphics tablets (digitizers). Most such devices produce online data, i.e. provide both spatial and temporal information about pen (stylus) position. As our approach requires spatial information

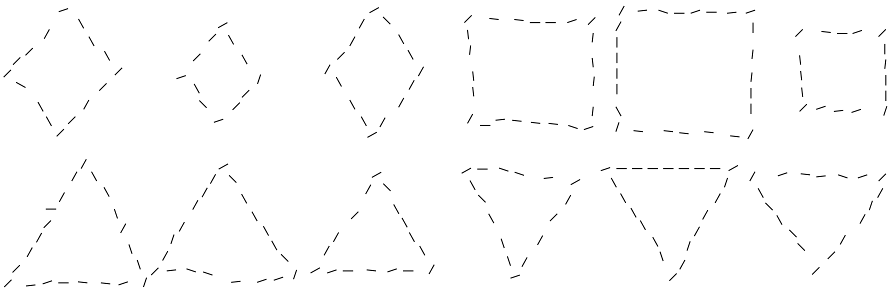


Fig. 4 Visualization of primitives derived from objects depicted in Fig. 3

only, its applicability spans also off-line interpretation of sketches stored as ordinary raster images (e.g. acquired from a paper drawing using a scanner).

According to Krishnapuram *et al.* [22], the complete task of sketch interpretation may be subdivided into three subtasks: (i) segmentation of drawing into disjoint shapes that are recognized independently, (ii) fitting of shapes (models) to drawings and (iii) recognition of particular shapes. The following experimental study tackles two latter tasks.

5.1 Training Data and Parameter Setup

Using a TabletPC computer, we prepared a training set containing 48 images of four elementary shapes: diamonds (D), rectangles (R), triangles pointing upwards (TU) and triangles pointing downwards (TD), each shape represented by 12 examples. The shapes were of different dimensions and orientations and were placed at random locations in a raster image of 640×480 pixels. Figure 3 illustrates selected training examples, shown for brevity in one image; note, however, that each shape is a separate training example. Figure 4 shows the visualization of primitives obtained from objects from Fig. 3. Each segment depicts a single VP, with its spatial coordinates located in the middle of the segment and the orientation depicted by slant.

To evolve our recognizers, we used generational evolutionary algorithm comprising a population of 25,000 GP individuals for 300 generations. Koza's [3] ramped half-and-half operator with ramp from 2 to 6 was used to produce the initial population. We applied tournament selection with a tournament of size 5, using individuals' sizes for tie breaking and thus promoting smaller GP trees. Offspring were created by crossing over selected parent solutions from previous generation (with probability 0.8) or mutating selected solutions (with probability 0.2). The GP tree depth limit was set to 10; the mutation and crossover operations might be repeated up to 5 times if the resulting individuals do not meet this constraint; otherwise, the parent solutions are copied into the subsequent generation. Except for the fitness function implemented for efficiency in C++, the algorithm has been implemented in Java with the help of the ECJ package [28]. For evolutionary parameters not mentioned here explicitly, ECJ's defaults have been used.

5.2 The Results

The experiment was conducted according to the following procedure. First, for each class of shape (D, R, TU and TD), five evolutionary runs were performed using the training set for fitness computation. From each run, the best individual (learner) was chosen. These 20 individuals constituted the *pool* of individuals, which we used to build a recognition system that was subsequently evaluated on a separate test set containing 124 shapes. We present results for two types of recognition systems: simple recognition system and voting recognizer.

(1) Simple recognition system consists of four best-in-class individuals, selected from the pool according to the fitness value (computed on the training data). This

Table 2 Test-set confusion matrices for simple recognition system (a) and voting recognition system (b) (rows: actual object classes, columns: system's decisions)

(a)

	D	R	TU	TD
D	**33**	0	3	0
R	1	**25**	3	2
TU	0	0	**28**	0
TD	5	0	0	**24**

(b)

	D	R	TU	TD
D	**34**	1	1	0
R	2	**27**	1	1
TU	0	0	**28**	0
TD	0	0	0	**29**

straightforward system performs recognition of the test example *t* by computing fitnesses of all four individuals for *t* and indicating the class associated with the fittest individual. The rationale behind such a procedure is as follows. As the learner was trained to perform well on images from one class, its fitness should be near 0 also for test images representing this class. For shapes coming from other classes, the learner will most probably fail to perform their reproduction, issuing inappropriate drawing actions and/or using an incorrect number of them. For example, it is unlikely that an individual that learned the concept of triangle could recognize squares; when applied to such an image, it will most likely receive a high (inferior) fitness value.

The simple recognition system achieves test-set accuracy of classification of 88.71%. The detailed confusion matrix is presented in Table 2a.

(2) Voting recognizer uses all 20 individuals from the pool and runs multiple votings to benefit from the diversity of individuals obtained from different evolutionary runs. Technically, all 5^4 possible combinations of individuals–voters are considered, always using one voter per class. Each voting produces one decision, using the same procedure as in the simple recognition system. The class indicated most frequently in all votings is the final decision made by the recognition system.

The voting recognizer performs significantly better than the simple recognition system and attains a test-set recognition accuracy of 95.16%. Table 2b presents its confusion matrix. Note that this approach completely eliminated the problem of confusing the TD class with the D class, which seemed to be the most challenging for the simple recognition system.

To get some reference point, we also performed comparative experiments with standard pattern recognition and machine learning techniques. For this purpose, we digitized our shapes to 32×32 black-and-white raster images. Next, we trained and tested a radial basis function network, a support vector machine with linear kernels, and a three-layer perceptron on these data, using default parameter settings provided by Weka software environment [29]. They attained test-set accuracy of 95.97%, 97.58%, and 97.58%, respectively. Thus, we can conclude that at least the voting recognizer's accuracy of classification is comparable to these standard algorithms.

In Fig. 5, we illustrate the outcomes of generative shape restoration, i.e., the canvases produced by the well-performing individuals for randomly selected test shapes. Thin dotted lines mark the shapes drawn by a human (input image *s*), whereas thick continuous lines depict drawing actions performed by the individual (sections in the canvas *c*). Darker lines reflect overlapping of multiple DAs. It

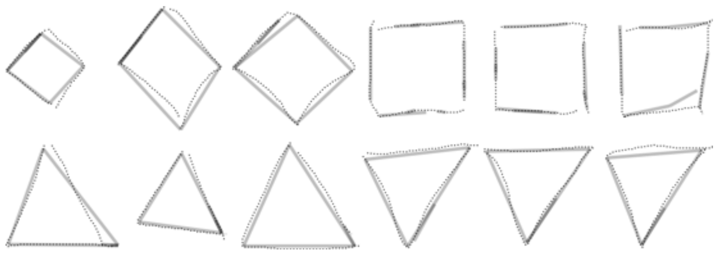


Fig. 5 The generative restoration process performed by trained learners

may be easily observed that, in most cases, the evolved individuals successfully reproduce the overall shape of the recognized object. Reproduction seems to be robust despite various forms of imperfectness of the hand-drawn figures.

5.3 Interpretation of the Evolved GP Code

To analyze the behaviour of an evolved recognition procedure, let us consider an exemplary well-performing individual trained to recognize objects from the TU class, presented in Fig. 6. This individual uses only one type of terminal nodes, *ImageNode*, which fetches the original input image; apparently, it copes well with the recognition problem without referring to other terminal symbols such as scalar constants (see Table 1). Several internal nodes with names ending with capital letter D are used to perform the drawing actions.

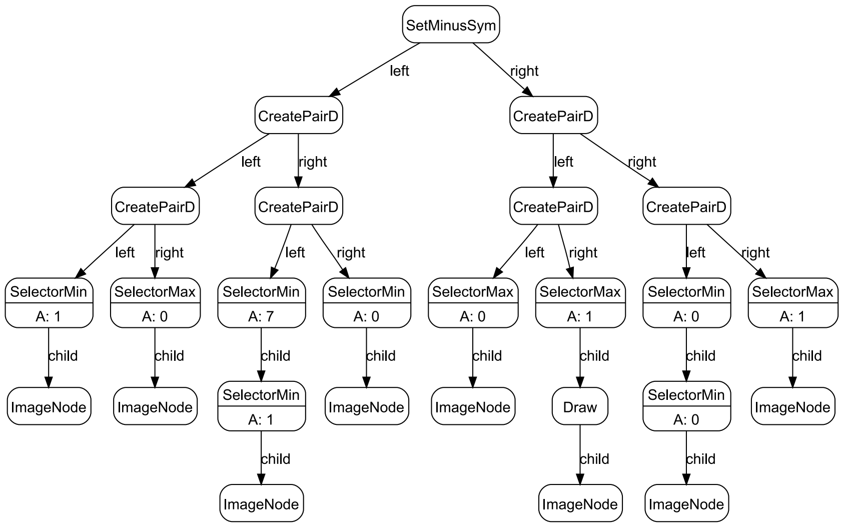


Fig. 6 An exemplary individual evolved for the TU class

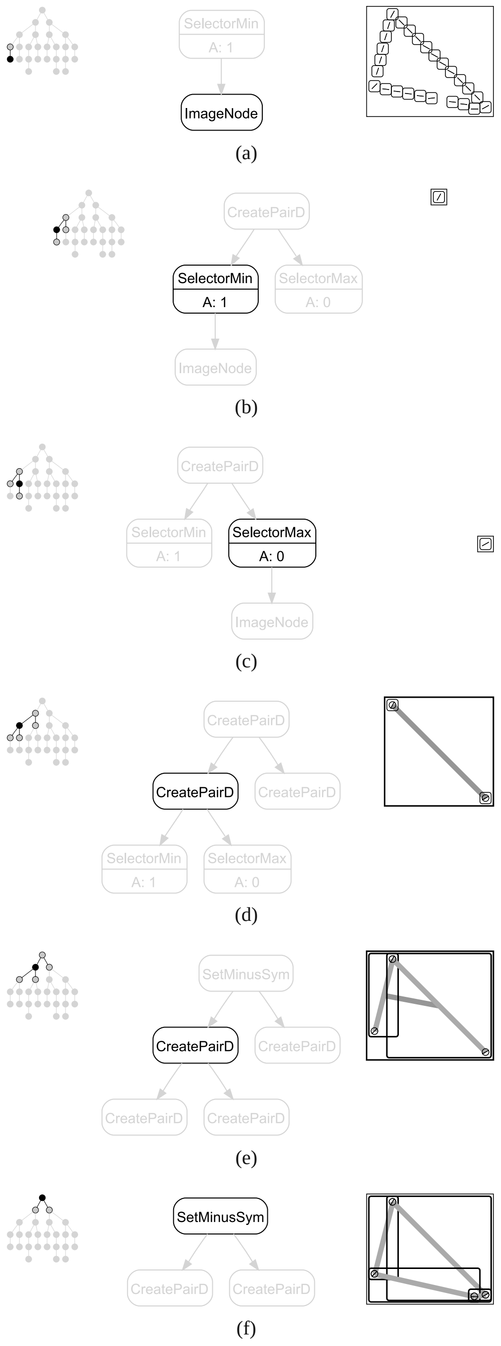


Fig. 7 Illustration of processing performed by individual from Fig. 6 for an exemplary triangular shape

In the following, we show how the individual presented in Fig. 6 exploits the generative and compositional aspects of our approach, which allow it to gradually restore the shape of the recognized object using primitive components (sections). In Fig. 7, we illustrate the shape restoration process taking place in this individual for a particular input image representing the TU class. Each subfigure (7a–f) corresponds to a specific stage of processing, marked in the miniature reproduction of Fig. 6 placed on the left-hand side. The middle part of each subfigure magnifies that view, presenting in detail the nodes the subfigure refers to. Finally, the right-hand part of each subfigure illustrates the output produced by the considered tree fragment, or, more precisely, by the node marked in black. The VPs are depicted as short segments, whereas sets of VPs are shown as rectangular frames; in particular, the hierarchical nesting of VP sets is illustrated by nesting appropriate rectangles.

In Fig. 7(a), we observe the *ImageNode* function returning the input set P of VPs. The VPs were derived from the input raster input image representing a triangular shape from the TU class. Each VP is depicted by a separate short segment, where the midpoint coordinates mark the location of the VP and slant visualizes the VP's orientation. Figure 7(b) shows how these data are transformed by the *SelectorMin* function. This function essentially boils down to $\arg\min_y(P)$, i.e. it selects from P the VP having the minmal value of the y coordinate (the y coordinate has #1 on the list of VP's attributes). As a result, *SelectorMin* returns a single VP located in the upper-left corner of the field of view (the origin of the coordinate system is in the upper-left corner).

Analogously, a sibling branch of the tree, detailed in Fig 7(c), filters the VP with the maximal value of coordinate x. Next, the *CreatePairD* node depicted in Fig. 7(d) combines the results produced by the *SelectorMin* and *SelectorMax* branches, grouping them into a new set, marked as an extra frame encompassing both VPs in the right-hand inset. However, more importantly, *CreatePairD* belongs to the group of drawing nodes, and so as a side effect, it draws a section spanning both VPs on the canvas c (superimposed on the input image). The drawn section becomes a part of individual's response to the input image and contributes (positively or negatively) to its fitness. Figure 7(e,f) illustrates how the other branches of the GP tree reconstruct the remaining parts of the input image on the canvas c, leading to a minimal (in the sense of number of used sections) and close-to-perfect restoration of the original triangular shape.

6 Conclusions

The obtained results demonstrate the ability of our generative visual learning to evolve object recognition systems that successfully classify shapes in hand-drawn sketches. The generative trait of the method does not allow the evolving learners to 'skim' the training data for *any* discriminating feature, but forces them to reproduce the training pattern, and thus, in a sense, to understand it. This, in turn, implies high performance on the test set despite limited amount of training data (12 examples per class only).

The evolved visual learners perform well, even though their background knowledge is very limited and encoded exclusively in general GP functions. We use the same set of functions as in our preliminary work on this topic [25, 26], which involved different shape classes; the function set has not been tuned to this specific training data. A parallel work in progress on handwritten character recognition attains competitive results using the same function set, thus proving its universal character.

Let us also emphasize the practical virtues of the proposed approach. First, our learners do not need negative examples. Each learner is trained only on examples from its own class (so-called *one-class learning*). This makes the obtained recognition systems easily extensible: to expand the set of recognized classes by another class, only one new evolutionary process has to be run, while the former individuals that make up the recognition system do not need to be modified. Second, the method operates at low time complexity. Recognition of an object takes on average only 100 ms when running on 2.4 GHz AMD Opteron processor. Processing carried out by the GP tree takes only a tiny fraction of that time, while most of it is spent on extraction of VPs from the raster image. For this reason, we extract VPs prior to the evolutionary run and store them in a cache memory. This, in turn, allows us to evolve large populations of individuals for a remarkable number of generations in a reasonable time.

In this preliminary study, we focused on recognition of basic geometrical shapes. Interpretation of more complex images may require more sophisticated and substantially larger GP individuals. To alleviate scalability issues that may arise in such a case, we devised an extended approach that performs automatic decomposition of image processing into multiple GP trees and enables sharing of decomposed trees between multiple learning tasks. Preliminary results indicate that such an approach is profitable in terms of convergence speed [25, 26]. Another future research direction could concern other aspects of visual information, such as colour or texture, or even their integration within one visual learner, and other image representations, such as region adjacency graphs.

Acknowledgement. This work has been supported by the Ministry of Science and Higher Education grant N N519 3505 33.

References

1. Bhanu, B., Lin, Y., Krawiec, K.: Evolutionary Synthesis of Pattern Recognition Systems. Springer, New York (2005)
2. Krawiec, K., Bhanu, B.: Visual learning by coevolutionary feature synthesis. IEEE Transactions on System, Man, and Cybernetics – Part B 35(3), 409–425 (2005)
3. Koza, J.R.: Genetic programming – 2. MIT Press, Cambridge (1994)
4. Banzhaf, W., Nordin, P., Keller, R.E., Francone, F.D.: Genetic Programming – An Introduction; On the Automatic Evolution of Computer Programs and its Applications. Morgan Kaufmann, San Francisco (1998)
5. Goldberg, D.: Genetic algorithms in search, optimization and machine learning. Addison-Wesley, Reading (1989)

6. Michalewicz, Z.: Genetic algorithms + data structures = evolution programs. Springer, Berlin (1994)
7. Koza, J.R.: Human-competitive applications of genetic programming. In: Ghosh, A., Tsutsui, S. (eds.) Advances in Evolutionary Computing: Theory and Applications, pp. 663–682. Springer, Berlin (2003)
8. Tackett, W.A.: Genetic programming for feature discovery and image discrimination. In: Forrest, S. (ed.) Proceedings of the 5th International Conference on Genetic Algorithms, ICGA 1993, University of Illinois at Urbana-Champaign, pp. 303–309. Morgan Kaufmann, San Francisco (1993)
9. Johnson, M.P., Maes, P., Darrell, T.: Evolving visual routines. In: Brooks, R.A., Maes, P. (eds.) ARTIFICIAL LIFE IV, Proceedings of the fourth International Workshop on the Synthesis and Simulation of Living Systems, pp. 198–209. MIT, Cambridge (1994)
10. Daida, J.M., Bersano-Begey, T.F., Ross, S.J., Vesecky, J.F.: Computer-assisted design of image classification algorithms: Dynamic and static fitness evaluations in a scaffolded genetic programming environment. In: Koza, J.R., Goldberg, D.E., Fogel, D.B., Riolo, R.L. (eds.) Genetic Programming 1996: Proceedings of the First Annual Conference, Stanford University, CA, USA, pp. 279–284. MIT Press, Cambridge (1996)
11. Winkeler, J.F., Manjunath, B.S.: Genetic programming for object detection. In: Koza, J.R., Deb, K., Dorigo, M., Fogel, D.B., Garzon, M., Iba, H., Riolo, R.L. (eds.) Genetic Programming 1997: Proceedings of the Second Annual Conference, Stanford University, CA, USA, pp. 330–335. Morgan Kaufmann, San Francisco (1997)
12. Rizki, M.M., Zmuda, M.A., Tamburino, L.A.: Evolving pattern recognition systems. IEEE Transactions on Evolutionary Computation 6(6), 594–609 (2002)
13. Howard, D., Roberts, S.C., Brankin, R.: Evolution of ship detectors for satellite SAR imagery. In: Langdon, W.B., Fogarty, T.C., Nordin, P., Poli, R. (eds.) EuroGP 1999. LNCS, vol. 1598, pp. 135–148. Springer, Heidelberg (1999)
14. Olague, G., Puente, C.: The honeybee search algorithm for three-dimensional reconstruction. In: Rothlauf, F., Branke, J., Cagnoni, S., Costa, E., Cotta, C., Drechsler, R., Lutton, E., Machado, P., Moore, J.H., Romero, J., Smith, G.D., Squillero, G., Takagi, H. (eds.) EvoWorkshops 2006. LNCS, vol. 3907, pp. 427–437. Springer, Heidelberg (2006)
15. Teller, A., Veloso, M.: PADO: A new learning architecture for object recognition. In: Ikeuchi, K., Veloso, M. (eds.) Symbolic Visual Learning, pp. 81–116. Oxford University Press, Oxford (1996)
16. Poli, R.: Genetic programming for image analysis. In: Koza, J.R., Goldberg, D.E., Fogel, D.B., Riolo, R.L. (eds.) Genetic Programming 1996: Proceedings of the First Annual Conference, Stanford University, CA, USA, pp. 363–368. MIT Press, Cambridge (1996)
17. Howard, D., Roberts, S.C.: Evolving object detectors for infrared imagery: a comparison of texture analysis against simple statistics. In: Miettinen, K., Makela, M.M., Neittaanmaki, P., Periaux, J. (eds.) Evolutionary Algorithms in Engineering and Computer Science, Jyvaskyla, Finland, pp. 79–86. John Wiley & Sons, Chichester (1999)
18. Lett, M., Zhang, M.: New fitness functions in genetic programming for object detection. In: Pairman, D., North, H., McNeill, S. (eds.) Proceeding of Image and Vision Computing International Conference, Akaroa, New Zealand, Lincoln, Landcare Research, pp. 441–446 (2004)
19. Krawiec, K.: Pairwise comparison of hypotheses in evolutionary learning. In: Brodley, C., Pohoreckyj-Danyluk, A. (eds.) Proc. Eighteenth International Conference on Machine Learning, pp. 266–273. Morgan Kaufmann, San Francisco (2001)

20. Krawiec, K.: Learning high-level visual concepts using attributed primitives and genetic programming. In: Rothlauf, F., Branke, J., Cagnoni, S., Costa, E., Cotta, C., Drechsler, R., Lutton, E., Machado, P., Moore, J.H., Romero, J., Smith, G.D., Squillero, G., Takagi, H. (eds.) EvoWorkshops 2006. LNCS, vol. 3907, pp. 515–519. Springer, Heidelberg (2006)
21. Revow, M., Williams, C.K.I., Hinton, G.E.: Using generative models for handwritten digit recognition. IEEE Transactions on Pattern Analysis and Machine Intelligence 18(6), 592–606 (1996)
22. Krishnapuram, B., Bishop, C.M., Szummer, M.: Generative models and bayesian model comparison for shape recognition. In: IWFHR 2004: Proceedings of the Ninth International Workshop on Frontiers in Handwriting Recognition (IWFHR 2004), pp. 20–25. IEEE Computer Society, Washington (2004)
23. Langley, P.: Elements of machine learning. Morgan Kaufmann, San Francisco (1996)
24. Kohavi, R., John, G.H.: Wrappers for feature subset selection. Artificial Intelligence Journal 2, 273–324 (1997)
25. Jaśkowski, W.: Genetic programming with cross-task knowledge sharing for learning of visual concepts. Master's thesis, Poznan University of Technology, Poznań, Poland (2006), `http://www.cs.put.poznan.pl/wjaskowski/pub/papers/jaskowski06crosstask.pdf`
26. Wieloch, B.: Genetic programming with knowledge modularization for learning of visual concepts. Master's thesis, Poznan University of Technology, Poznań, Poland (2006)
27. Krawiec, K.: Generative learning of visual concepts using multiobjective genetic programming. Pattern Recognition Letters 28(16), 2385–2400 (2007)
28. Luke, S.: ECJ evolutionary computation system (2002), `http://cs.gmu.edu/eclab/projects/ecj/`
29. Witten, I.H., Frank, E.: Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations. Morgan Kaufmann, San Francisco (1999)

Optimizing a Medical Image Analysis System Using Mixed-Integer Evolution Strategies

Rui Li, Michael T.M. Emmerich, Jeroen Eggermont, Ernst G.P. Bovenkamp, Thomas Bäck, Jouke Dijkstra, and Johan H.C. Reiber

Abstract. We will discuss Mixed-Integer Evolution Strategies (MIES) and their application to the optimization of control parameters of a semi-automatic image analysis system for Intravascular Ultrasound (IVUS) images. IVUS is a technique used to obtain real-time high-resolution tomographic images from the inside of coronary vessels and other arteries. The IVUS image feature detectors used in the analysis system are expert-designed and the default parameters are calibrated manually so far. The new approach, based on MIES, can automatically find good parameterizations for sets of images, which provide in better results than manually tuned parameters. From the algorithmic point of view, the difficulty is designing a black-box optimization strategy that can deal with nonlinear functions and different types of parameters, including integer, nominal discrete and continuous variables. MIES turns out to be well suited for this task. The results presented in this contribution will summarize and extend recent studies on benchmark functions and the IVUS image analysis optimization problem.

Keywords: Intravascular Ultrasound (IVUS), Evolution Strategies (ES), Mixed-Integer Evolution Strategies (MIES), Coronary Artery Disease (CAD).

1 Introduction

Feature detection in medical images is a key technology in the medical field. Often, complex and variable structures, such as calcified plaque in arteries, are to be

Rui Li, Michael T.M. Emmerich, and Thomas Bäck
Natural Computing Group, Leiden Institute of Advanced Computer Science,
Leiden University, The Netherlands
e-mail: {ruili,emmerich,baeck}@liacs.nl

Jeroen Eggermont, Ernst G.P. Bovenkamp, Jouke Dijkstra, and Johan H.C. Reiber
Division of Image Processing, Department of Radiology C2S,
Leiden University Medical Center, The Netherlands
e-mail: {J.Eggermont,E.G.P.Bovenkamp,J.Dijkstra}@lumc.nl
J.H.C.Reiber@lumc.nl

detected and modelled in images or sequences of images. The development of feature detection systems has received much attention in medical and computer science research. However, the performance of most systems depends on a large number of control parameters, and the setting of these control parameters is usually done by means of an educated guess or manual tuning using trial and error.

In this contribution, we argue that manual tuning is often not sufficient to exploit the full potential of image detection systems, i.e. it leads to suboptimal parameter settings. We propose a versatile and robust procedure for automated parameter tuning based on Mixed-Integer Evolution Strategies (MIES). Compared to the manual trial-and-error approach, with MIES, the systems developer can search for optimized parameter settings automatically and will likely obtain parameter settings that lead to significant higher accuracy of the feature detectors.

Among other image acquisition techniques, Intravascular Ultrasound (IVUS) received major attention for analyzing the structure of coronary blood vessels. Due to noise, pullback movements of the catheter and the variability of structures, even for human experts, it can be difficult to interpret IVUS image sequences. Therefore, the development of tailored computer-assisted image analysis has received major attention in recent years [1–3].

However, today's methods, directed at the automated recognition of certain structures in images, are applicable only over a limited range of standard situations. To overcome this problem, an image interpretation system, based on the paradigm of multiagents [1, 4], using the cognitive architecture Soar [5], was successfully developed over the past years. Agents in this system dynamically adapt their segmentation algorithms. This adaptation is based on knowledge about global constraints, contextual knowledge, local image information and personal beliefs such as confidence in their own image processing results. Although, in practice, the multiagent system has showed to offer lumen and vessel detection comparable to human experts [1], it is designed for symbolic reasoning, not numerical optimization. Besides, it is almost impossible for a human expert to completely specify how an agent should adjust its feature detection parameters in each and every possible interpretation context. As a result, an agent has only control knowledge for a limited number of contexts and a limited set of feature detector parameters. This knowledge has to be updated whenever something changes in the image acquisition pipeline. Therefore, it would be much better if such knowledge might be acquired by learning the optimal parameters for different interpretation contexts automatically.

This contribution addresses the problem of learning these optimal parameter settings from a set of example segmentations. It is an optimization problem that is difficult to solve in practice with standard numerical methods (such as gradient-based strategies), as it incorporates different types of parameters, and confronts the algorithms with a complex geometry (rugged surfaces and discontinuities). Moreover, the high dimensionality of this problem makes it almost impossible to find optimal settings through manual experimentation.

Encouraged by previous work [6, 7] on optimization of image segmentation algorithms in the medical domain and other application fields, we consider MIES as a solution method. Unlike these previous approaches, MIES are more suitable for dealing with continuous parameters and can handle difficult mixed-integer parameter optimization problems as encountered in the image processing domain.

The chapter is structured as follows: First, in Sect. 2, we explain the application of IVUS image analysis in more detail. Then, in Sect. 3, MIES (MIES) are introduced. In Sect. 4, MIES are tried on a set of artificial problems and compared to standard evolution strategies (ES). MIES are then applied to the parameter optimization of an IVUS image analysis system in Sect. 5. Finally, conclusions and future work are presented in Sect. 6.

2 Intravascular Ultrasound Image Analysis

Cardiovascular disease is the leading cause of death in the United States and coronary artery disease has the highest percentage (53%) of death among the heart diseases according to the American Heart Association Heart Disease and Stroke Statistics [8].

Atherosclerosis is a disease characterized by a deposit of plaque in an arterial wall over time. The disruption of an atherosclerotic plaque is considered to be the most frequent cause of heart attack and sudden cardiac death. Studying vulnerable plaques constitutes a major research area in the field of clinical and medical imaging.

IVUS provides real-time high-resolution tomographic images of the coronary vessels wall and is able to show the presence or absence of compensatory artery enlargement. IVUS allows precise tomographic measurement of the lumen area and plaque size, distribution and, to some extent, composition of the plaque. An example of an IVUS image is shown in Fig. 1.

To obtain insight into the status of an arterial segment, a so-called catheter pullback is carried out: an ultrasound probe is positioned distally (downstream) of the

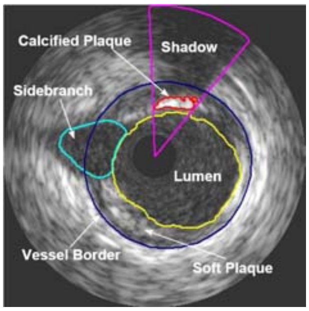


Fig. 1 An IVUS image with detected features. The *black circle* in the middle is where the ultrasound imaging device (catheter) was located. The *dark area* surrounding the catheter is called the *lumen*, which is the part of the artery where the blood flows. Above the catheter, *calcified plaque* is detected, which blocks the ultrasound signal causing a dark *shadow*. Between the inside border of the vessel and the lumen, there is some soft plaque, which does not block the ultrasound signal. The *dark area* left of the catheter is a *sidebranch* (another vessel)

segment of interest and then mechanically pulled back (today typically at a speed of 0.5 mm/s) during continuous image acquisition to the proximal (upstream) part of the segment of interest. Experienced users may then conceptualize the complex 3D structure of the morphology and pathology of the arterial segment from this stack of images by reviewing such a sequence repeatedly. Typically, one such pullback sequence consists of 500–1000 images, which represents about 50 mm of vessel length.

As we can see from Fig. 1, IVUS images contain image artifacts, drop-out regions and different kinds of tissue. Furthermore, manual segmentation of IVUS images is very time consuming and highly sensitive to intra- and inter-observer variability [1], while the data sets are very large. This makes IVUS image analysis a non-trivial medical application domain where a sophisticated image interpretation approach is warranted.

3 Mixed-Integer Evolution Strategies

Evolution Strategies (ES) [9–11] are robust global optimization strategies that are frequently used for continuous optimization. They belong to the class of randomized search heuristics and use principles of organic evolution, such as selection, recombination and mutation. As a flexible and robust search technique, they have been successfully applied to various real-world problems.

Here, we use a special variant of an ES for the simultaneous optimization of continuous, integer and nominal discrete parameters. This variant, the so-called MIES, was introduced in [7]. It combines mutation operators of evolution strategies in the continuous domain [11], for integer programming [12] and for binary search spaces [9]. These operators have certain common desirable properties, such as symmetry, scalability and maximal entropy, the details of which will be discussed later. The MIES was originally developed for optical filter optimization [9, 13] and chemical engineering plant optimization [7, 14]. Recently, as discussed in this contribution, it has been used in the context of medical image analysis [15, 16]. In the latter work, as well as in [7], its convergence behaviour on various artificial landscapes was also studied empirically, including a collection of single-peak landscapes in [7] and landscapes with multiple peaks in [15, 16].

3.1 Search Space and Problem Definition

A general, formal problem statement for the mixed integer global optimization problems that we will consider in this contribution reads:

$$f(r_1,\ldots,r_{n_r},z_1,\ldots,z_{n_z},d_1,\ldots,d_{n_d}) \rightarrow \min \qquad (1)$$

subject to

$$\begin{aligned} r_i \in R_i &= [r_i^{min}, r_i^{max}] \subset \mathbb{R},\ i = 1, \ldots, n_r \\ z_i \in Z_i &= [z_i^{min}, z_i^{max}] \subset \mathbb{Z},\ i = 1, \ldots, n_z \\ d_i \in D_i &= \{d_{i,1}, \ldots, d_{i,l_i}\}, i = 1, \ldots, n_d \end{aligned}$$

Here, three types of variables occur:

Continuous Variables, denoted with r_i, are taken from an interval $R_i \subset \mathbb{R}$ and their values are represented as floating point numbers. In the image processing field, for instance, threshold parameters or a radius parameter for a geometrical shape are often represented as continuous variables.

Integer Variables, denoted with z_i, are taken from a range of integer variables $Z_i \subset \mathbb{Z}$. Important characteristics of integer variables are that their values have a smallest neighborhood (as opposed to continuous variables) and that a linear ordering is defined on the values (as opposed to nominal discrete variables). The number of grey values in an image is a typical example of an integer variable in the image processing domain.

Nominal Discrete Variables, denoted with d_i, are variables whose values are taken from a finite domain, denoted with D_i. Neither a metric nor an ordering is defined on this domain. An example is a variable that takes its value from a set of geometrical shapes (ellipse, square and triangle). Also, binary variables (such as switches) belong to this class of variables.

The objective function f is considered to be a black-box function, though we assume that similar input parameter vectors result in similar function values. We do not mention constraints explicitly in the problem definition and assume that they are integrated in the formulation of f by means of a metric penalty [17].

3.2 Algorithm Description

The problem of designing an evolution strategy for a new type of search space breaks down into three subtasks: (1) definition of the generational cycle, (2) definition of the individual representation and (3) definition of variation operators for the representation of choice. These subtasks will be discussed next.

The chosen algorithm will be an instantiation of a $(\mu \overset{+}{,} \lambda)$-ES for mixed-integer spaces. It generalizes the more common $(\mu \overset{+}{,} \lambda)$-ES for continuous spaces, the dynamic behaviour of which was subject to thorough theoretical and empirical studies. For instance, Schwefel [11] compared it to traditional direct optimization algorithms. Theoretical studies of the convergence behaviour of the ES were carried out, for instance, by Beyer [18] and Rudolph [19]. A comparison to other evolutionary algorithms such as genetic algorithms can be found in Bäck [9]. The results indicate that the ES is a robust optimization tool that can deal with a large number of practically relevant function classes, including discontinuous and multimodal functions. In addition, the ES performance scales well with the search space dimension.

Generational cycle

The main procedure of the MIES is described in Algorithm 1. After a random initialization and evaluation of the first population $P(0)$ of μ individuals (parameter vectors taken from an individual space I) and setting the generation counter t to zero, the iteration starts. In a first step of the iteration, the algorithm generates the set $Q(t)$ of λ new offspring individuals, each of them obtained by the following procedure: Two individuals are randomly selected from $P(t)$ and an offspring is generated by recombining these parents and then mutating (random perturbation) the individual resulting from the recombination. In the next step, the λ offspring individuals are evaluated using the objective function to rank the individuals (the lower the objective function value the better the rank). In case of a $(\mu + \lambda)$ selection, the μ best individuals out of the union of the λ offspring individuals and the μ parental individuals are selected. In case of a (μ, λ) selection, the μ best individuals out of the λ offspring individuals are selected. The selected individuals form the new parent population $P(t+1)$. After this, the generation counter is incremented. The generational loop is repeated until the termination criterion[1] is fulfilled.

Algorithm 1. $(\mu \overset{+}{,} \lambda)$-Evolution Strategy

```
$t \leftarrow 0$
**initialize** Population $P(t) \in I^{\mu}$
**evaluate** the $\mu$ initial individuals with objective function $f$
**while** Termination criteria not fulfilled **do**
    **for all** $i \in \{1, \ldots, \lambda\}$ **do**
        choose randomly $c_{i_1}$ and $c_{i_2}$ from $P(t)$ (repetition is possible)
        $x_i \leftarrow$ mutate(recombine($c_{i_1}, c_{i_2}$))
        $Q(t) \leftarrow Q(t) \cup \{x_i\}$
    **end for**
    $P(t+1) \leftarrow \mu$ individuals with best objective function value from $P(t) \cup Q(t)$ (plus), or
    $Q(t)$ (comma)
    $t \leftarrow t+1$
**end while**
```

Representation

An individual in an evolution strategy contains the information about one solution candidate. The contents of individuals are inherited by future individuals and is subject to variation. The standard representation of a solution in an ES individual is a continuous vector. In addition, parameters of the probability distribution used in the mutation (such as standard deviations or 'step-sizes') are stored in the individual. The latter parameters are referred to as strategy parameters.

[1] In most cases, a maximal number of generations is taken as termination criterion.

To solve mixed-integer problems with an evolution strategy, we extend the real-vector representations of individuals by introducing integer and discrete variables as well as strategy parameters related to them. The domain of individual then reads:

$$I = R_1 \times \cdots \times R_{n_r} \times Z_1 \times \cdots \times Z_{n_z} \times D_1 \times \cdots \times D_{n_d} \times A_s$$

Here, A_s denotes the domain of strategy parameters and is defined as

$$A_s = \mathbb{R}_+^{n_\sigma + n_\varsigma} \times [0,1]^{n_p}, n_\sigma \leq n_r, n_\varsigma \leq n_z, n_p \leq n_d$$

An individual of a population *P(t)* in generation *t* is denoted as

$$\mathbf{a} = (r_1, \ldots, r_{n_r}, z_1, \ldots, z_{n_z}, d_1, \ldots, d_{n_d}, \sigma_1, \ldots, \sigma_{n_\sigma}, \varsigma_1, \ldots, \varsigma_{n_\varsigma}, p_1, \ldots, p_{n_p})$$

The so-called object variables, $r_1, \ldots, r_{n_r}, z_1, \ldots, z_{n_z}, d_1, \ldots, d_{n_d}$, determine the objective function value and thus the fitness of the individual. Here, $r_1, \ldots, r_{n_r}$ denote real valued, $z_1, \ldots, z_{n_z}$ integer valued and $d_1, \ldots, d_{n_d}$ nominal discrete variables. The so-called strategy variables $\sigma_1, \ldots, \sigma_{n_\sigma}$, are standard deviations used in the mutation of the real-valued variables, and $\varsigma_1, \ldots, \varsigma_{n_\varsigma}$ denote mean step sizes in the mutation of the integer parameters. Finally, $p_1, \ldots, p_{n_p}$ denote mutation probabilities (or rates) for the discrete object parameters. All these parameters are subject to inheritance, recombination and mutation within Algorithm 1.

Recombination

The recombination operator can be subdivided into two steps: selection of the parents and recombining the selected parents. Here, we will focus on local recombination which works with two recombination partners. The two recombination partners $c_1 \in I$ and $c_2 \in I$ are chosen randomly, uniformly distributed, from the parental generation for each of the offspring individuals. The information contained in these individuals is combined in order to generate an offspring individual. In evolution strategies, two recombination types are commonly used: dominant and intermediate recombination [11]. In a *dominant recombination*, the operator randomly chooses one of the corresponding parental parameters for each offspring vector position. *Intermediate recombination* computes the arithmetic mean of both parents and thus, in general, can only be applied for continuous object variables and strategy variables. In MIES, dominant recombination is used for the solution parameters, while intermediate recombination is used for the strategy parameters.

Mutation

For the parameter mutation, standard mutations with maximal entropy for real, integer and discrete parameter types are combined, as described in [9, 11–13]. The choice of mutation operators was guided by the following requirements for a mutation:

- *Accessibility:* Every point of the individual search space should be accessible from any other point by means of a finite number of applications of the mutation operator.
- *Feasibility:* The mutation should produce feasible individuals. This guideline can be crucial in search spaces with a high number of infeasible solutions.
- *Symmetry:* No additional bias should be introduced by the mutation operator.
- *Similarity:* Evolution strategies are based on the assumption that a solution can be gradually improved. This means it must be possible to generate similar solutions by means of mutation.
- *Scalability:* There should be an efficient procedure by which the strength of the impact of the mutation operator on the fitness values can be controlled.
- *Maximal Entropy:* If there is no additional knowledge about the objective function available the mutation distribution should have maximal entropy [12]. By this measure, a more general applicability can be expected.

Respecting these guidelines, the following operators have been selected in [7]:

The mutation of *continuous variables* is described in Algorithm 2. The new individual is obtained by adding a normal distributed random perturbation to the old values of the vector. The corresponding standard deviations are also subject to the evolution process and are thus multiplied in each step with a logarithmic distributed random number. Schwefel [11] termed the resulting process as *self-adaptive*, because of the capability of the process to adapt the step-sizes to the local fitness landscape. The general idea behind self-adaptation is that if a set of different individuals is generated, all with a different probability distribution, the individual with the best object variables is also likely the one with the best probability distribution that leads to the generation of these object variables. Thus, the parameters of this probability distribution are also inherited by the offspring individual.

We will now provide some remarks on the properties of the normal distributions, as they are responsible for the choice of this type of distribution for mutating continuous variables. Among all continuous distributions with finite variance on $\mathbb{R}$, the normal distribution possesses the maximum entropy. The multidimensional normal distribution is symmetrical to its mean value and is unimodal. The step-sizes represent standard deviations of the multidimensional normal distribution for each real-valued variable. By varying these standard deviations, the impact of the mutation on the variable vector in terms of similarity can be scaled.

As opposed to continuous variables, integer variables are less commonly used in evolution strategies. The mutation procedure for integer variables is borrowed from [12], where the replacement of the normal distribution by a geometrical distribution (G) has been suggested. Among distributions defined on integer spaces, the multidimensional geometric distribution is one of maximum entropy and finite variance. As the original geometric distribution is single-tailed, Rudolph [12] suggested to use instead the difference $Z_1 - Z_2$ of two geometrically distributed random variables. The resulting distribution is depicted for the 2-D case in Fig. 2. It is l_1-symmetrical[2] to its mean value, unimodal and has an infinite support. Thereby, symmetry and

[2] That is, symmetrical w.r.t. the manhattan distance.

Algorithm 2. Mutation of real-valued parameters

```
**input**: $r_1, \ldots, r_{n_r}, \sigma_1, \ldots, \sigma_{n_r}$
**output**: $r'_1, \ldots, r'_{n_\sigma}, \sigma'_1, \ldots, \sigma'_{n_\sigma}$
**control parameters:** $n_\sigma \in \{0,1\}$
$N_c \leftarrow N(0,1)$ {Generate and store a normal distributed random number}
$\tau \leftarrow \frac{1}{\sqrt{2*(n_r)}}; \tau' \leftarrow \frac{1}{\sqrt{2\sqrt{n_r}}}$ {Initialize global and local learning rate}
if $n_\sigma = 1$ {Single step-size mode} then
  $\sigma'_1 = \sigma_1 \exp(\tau N_c)$
  for all $i \in \{1, \ldots, n_r\}$ do
    $r'_i \leftarrow r_i + \sigma'_1 N(0,1))$
  end for
else
  {Multiple step-size mode}
  for all $i \in \{1, \ldots, n_r\}$ do
    $\sigma'_i \leftarrow \sigma_i \exp(\tau N_c + \tau' N(0,1))$
    $r'_i \leftarrow r_i + \sigma'_i N(0,1))$
  end for
end if
{Interval boundary treatment}
for all $i \in \{1, \ldots, n_r\}$ do
  $r'_i \leftarrow T_{[r_i^{min}, r_i^{max}]}(r'_i)$
end for
```

accessibility of the mutation is given. The strength of the mutation for the integer parameters is controlled by a set of step-size parameters, which represent the mean value of the absolute variation of the integer object variables. The details of this mutation operator are found in Algorithm 3. Note that a geometrically distributed random value can be generated by transforming a uniformly distributed random value u using:

$$z = \left\lfloor \frac{\ln(1-u)}{\ln(1-\psi)} \right\rfloor, \quad \psi = 1 - \varsigma \left(1 + \sqrt{1+\varsigma^2}\right)^{-1} \tag{2}$$

The width of the distribution can be controlled by the parameter ς. Excepting the different distribution types used, it is very similar to the real-valued mutation operator in Algorithm 2. Self-adaptation is used to control the width parameter(s). The mutation of the width parameter is done as in [12] using a global learning rate τ and local learning rate τ'.

Since we have to keep integer and continuous parameters in their feasible interval, the mutation operators need to be extended. Therefore, a transformation function that reflects parameters back into the feasible domain is applied in the mutation operators.

For the continuous and integer parameters, this is achieved by the *transformation function* $T_{[a,b]}$. The transformation function may be illustrated as a reflection at the interval boundaries. It can be implemented as illustrated in Algorithm 4. It cannot be

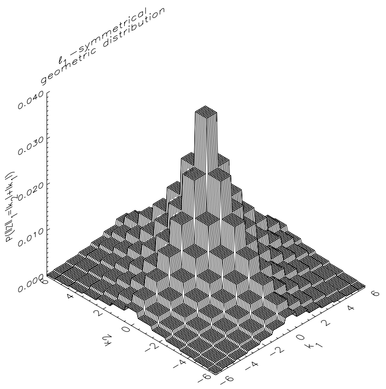


Fig. 2 A 2-D representation of the distribution obtained as the difference of two geometrical distributions. Figure courtesy of Günter Rudolph

avoided that a certain bias is introduced by the transformation function. However, keeping the ratio between step-size and interval width low, the main characteristics of the unconstrained mutation are preserved.

Algorithm 3. Mutation of integer parameters

```
input: $z_1, \dots, z_{n_z}, \varsigma_1, \dots, \varsigma_{n_\varsigma}$
output: $z_1, \dots, z_{n_z}, \varsigma_1, \dots, \varsigma_{n_\varsigma}$
control parameters: $n_\varsigma$
$N_c \leftarrow N(0,1)$
$\tau \leftarrow \frac{1}{\sqrt{2*(n_z)}}; \tau' \leftarrow \frac{1}{\sqrt{2\sqrt{n_z}}}$
if $n_\varsigma = 1$ then
    $\varsigma_1' \leftarrow \max(1, \varsigma_1 \exp(\tau N_c))$
    for all $i \in \{1, \dots, n_z\}$ do
        $u_1 \leftarrow U(0,1); u_2 \leftarrow U(0,1); \psi \leftarrow 1 - (\varsigma_1'/n_z)(1 + \sqrt{1 + (\frac{\varsigma_1'}{n_z})^2})^{-1}$
        $G_1 \leftarrow \left\lfloor \frac{\ln(1-u_1)}{\ln(1-\psi)} \right\rfloor; G_2 \leftarrow \left\lfloor \frac{\ln(1-u_2)}{\ln(1-\psi)} \right\rfloor$
        $z_i' \leftarrow z_i + G_1 - G_2$
    end for
else
    for all $i \in \{1, \dots, n_z\}$ do
        $\varsigma_i' \leftarrow \max(1, \varsigma_i \exp(\tau N_c + \tau' N(0,1)))$
        $u_1 \leftarrow U(0,1); u_2 \leftarrow U(0,1); \psi \leftarrow 1 - (\varsigma_i'/n_z)(1 + \sqrt{1 + (\frac{\varsigma_i'}{n_z})^2})^{-1}$
        $G_1 \leftarrow \left\lfloor \frac{\ln(1-u_1)}{\ln(1-\psi)} \right\rfloor; G_2 \leftarrow \left\lfloor \frac{\ln(1-u_2)}{\ln(1-\psi)} \right\rfloor$
        $z_i' \leftarrow z_i + G_1 - G_2$
    end for
end if
for all $i \in \{1, \dots, n_z\}$ do
    $z_i' \leftarrow T_{[z_i^{min}, z_i^{max}]}(z_i')$
end for
```

Finally, a mutation of the discrete parameters is carried out with a mutation probability as described in Algorithm 5. The probability is a strategy parameter for each discrete variable. Each new value is chosen randomly (uniformly distributed) out of the finite domain of values. The application of a uniform distribution is due to the principle of maximal entropy, since the assumption was made that there is no reasonable order defined between the discrete values. To reason about the requirements such as symmetry and scalability, we need to define a distance measure on the discrete sub-space. The assumption that there is no order, which can be defined on the finite domains of discrete values, leads to the application of the overlap distance measure: $\Delta((d_1,\dots,d_{n_d}),(d'_1,\dots,d'_{n_d})) = \sum_{i=1}^{n_d} H(d_i = d'_i)$ with $H(true)=1; H(false)=0$ as a similarity measure for guiding the design of the mutation operator.

A self-adaptation of the mutation probability for the discrete parameters is achieved by a logistic mutation of these parameters, generating new probabilities in the feasible domain. The logistic transformation function is recommended and discussed by Schütz and Sprave[13]. The basic idea of this transformation is to keep the variables within the range $[0,1]$. Given an original mutation probability $p \in [0,1]$, it can be mutated using the following procedure:

$$p' = \frac{1}{1+\frac{1-p}{p} * \exp(-\tau' N(0,1))} \tag{3}$$

Here, $N(0,1)$ denotes a function that returns a normally distributed random number. This mutation has the property that it is as likely to increase p than to decrease

Algorithm 4. Computation $T_{[a,b]}(x)$, for interval boundaries a and b.

```
$y = (x-a)/(b-a)$
**if** $\lfloor y \rfloor$ mod2 = 0 **then**
    $y' = |y - \lfloor y \rfloor|$
**else**
    $y' = 1 - |y - \lfloor y \rfloor|$
**end if**
$x' = a + (b-a)y'$
**return** $x'$
```

Algorithm 5. Mutation of nominal discrete parameters

```
**input:** $d_1,\dots,d_{n_d}, p_1,\dots,p_{n_p}$
**output:** $d'_1,\dots,d'_{n_d}, p'_1,\dots,p'_{n_p}$
**for all** $i \in \{1,\dots,n_p\}$ **do**
    $p'_i \leftarrow \frac{1}{1+\frac{1-p_i}{p_i}*\exp(-\tau' * N(0,1))}$
    $p'_i = T_{[0.01,0.5]}(p'_i)$
    **if** $U(0,1) < p'_i$ **then**
        choose a new element uniform distributed out of $D_i \setminus \{d_i\}$
    **end if**
**end for**
```

its value. To prevent that the mutation rate sticks to the boundaries, we recommend employing a second transformation function ($T_{[p_{min},p_{max}]}$) that keeps the value of p in the interval $[0.01, 0.5]$. The upper bound of 0.5 for the mutation probability is motivated by the observation that the mutation looses its causality once the probability exceeds the value of 0.5. The lower bound is to prevent the mutation probability from being too close to 0.

Depending on the discrete subspace, it can be advantageous to use a single mutation probability instead of many individual mutation probabilities $p_1, \ldots, p_{n_d}$. In case of a single mutation probability, for each position of the discrete subvector, it is decided independently, but with the same probability, whether to mutate this position or not.

4 Experiments on Artificial Test Problems

Some selected empirical results demonstrate the convergence behaviour of the proposed MIES algorithm. In this section, test results for theoretical test functions are illustrated. This gives the reader the opportunity to compare the results of this EA with those of other optimization algorithms, for instance, with standard ES. Since rather simple test functions have been chosen, the experiments give hints on the minimal running time for practical problems of similar dimension.

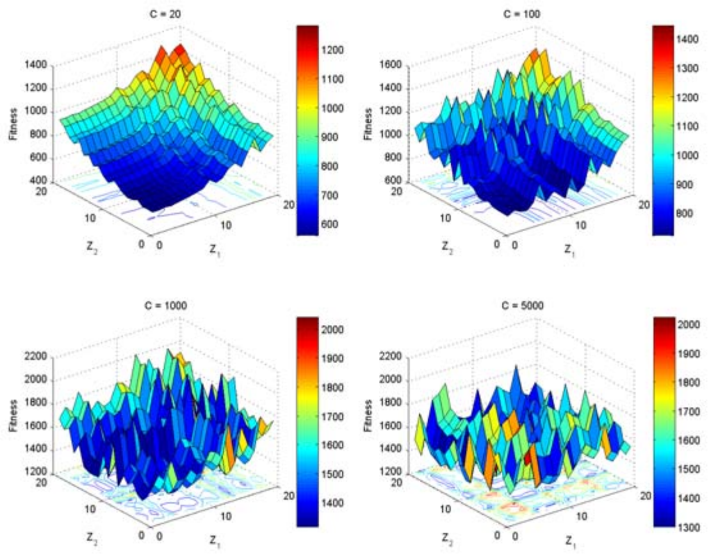


Fig. 3 Surface plots of the barrier test function for two integer variables Z_1 and Z_2, and the control parameter C = 20, 100, 1000 and 5000. All other variables were kept constant at a value of zero, Z_1 and Z_2 values were varied in the range from 0 to 19

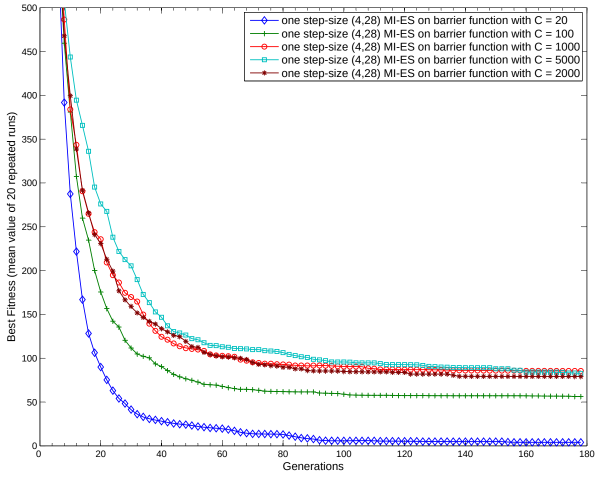


Fig. 4 The best results for (4,28) MIES using a single step-size on barrier functions with different C

4.1 Test Functions

We designed a multi-modal barrier problem generator that produces integer optimization problems with a scalable degree of ruggedness (determined by parameter C) by generating an integer array A using the following algorithm:

```
$\mathtt{A}[i] = i, i = 0,\ldots,20$
for $k \in \{1,\ldots,C\}$ do
    $j \leftarrow$ random number out of $\{0,\ldots,19\}$
    swap values of $\mathtt{A}[j]$ and $\mathtt{A}[j+1]$
end for
```

For $C = 0$, the ordering of the variable $y \in [0,20]$ values corresponds to the ordering of values of $A(y)$. If the value of C is slightly increased, only part of the order will be preserved under the mapping A, and thus similarity information can be exploited. Then, a barrier function is computed:

$$f_{\text{barrier}}(\mathbf{r},\mathbf{z},\mathbf{d}) = \sum_{i=1}^{n_r} \mathtt{A}[\lfloor r_i \rfloor]^2 + \sum_{i=1}^{n_z} \mathtt{A}[z_i]^2 + \sum_{i=1}^{n_d} \mathtt{B_i}[d_i]^2 \rightarrow \min \tag{4}$$

$$n_r = n_z = n_d = 5, \mathbf{r} \in [0,20]^{n_r} \subset \mathbb{R}^{n_r},$$
$$\mathbf{z} \in [0,20]^{n_z} \subset \mathbb{Z}^{n_z}, \mathbf{d} \in \{0,\ldots,20\}^{n_d} \subset \mathbb{R}^{n_d}. \tag{5}$$

Here, $B_i[0],\ldots,B_i[20]$ denotes a set of i permutations of the sequence $0,\ldots,20$, each of which being a random permutation fixed before the run. This construction prevents that the value of the nominal value d_i is quantitatively (anti-)correlated with

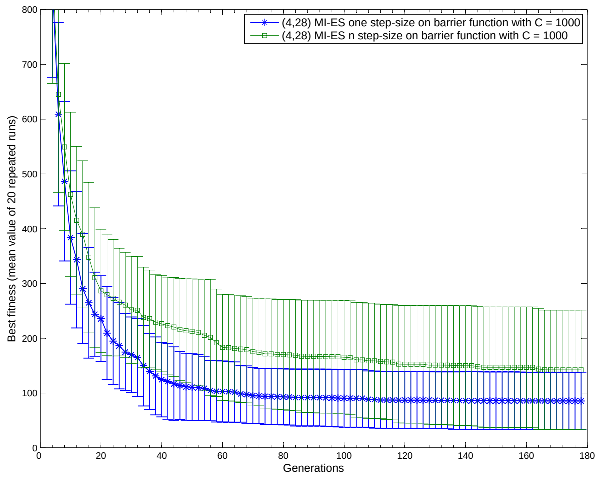


Fig. 5 Mean ± standard deviation for (4,28) MIES with different step-size modes on barrier function with $C = 1000$

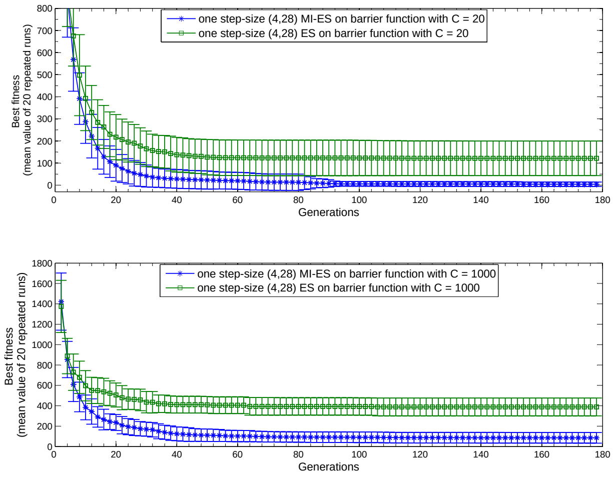


Fig. 6 Mean ± standard deviation for single step-size (4,28) MIES and ES on barrier functions with $C = 20$ and $C = 100$

the value of the objective function f. Such a correlation would contradict with the assumption that d_i are nominal values. Whenever a correlation between neighbouring values can be assumed, it is wiser to assign them to the ordinal type and treat them accordingly.

The parameter C controls the ruggedness of the resulting function with regard to the integer space. High values of C result in rugged landscapes with many barriers.

To get an intuition about the influence of C on the geometry of the function, we included plots for a two-variable instantiation of the barrier function in Fig. 3 for $C = 20, C = 100, C = 1000$ and $C = 5000$.

4.2 Experimental Results

The following MIES settings [16] are chosen for the experiments on the barrier problems: $(\mu = 4, \lambda = 28)$ for the population and offspring sizes and $(n_\sigma = n_\varsigma = n_p = 1)$ or $(n_\sigma = n_r, n_\varsigma = n_z, n_p = n_d)$ for the step-size mode. Since ES are stochastic algorithms, in the empirical experiments for each algorithm setting, we performed 20 runs in order to compute the mean value and standard deviation of the intermediate and results.

Figure 4 shows best fitness values[3] found by one step-size (4, 28) MIES on barrier functions with different control parameters C. It turns out that it is more difficult for (4,28) MIES to find global optima for barrier function with higher C value. Also, we noticed that when C is above a certain value, such as 1000, 2000 and 5000 in Fig. 4, the difficulty of the barrier function will not change too much.

From Fig. 5, we learn that it is more reliable with regard to the performance of the strategy to use a one step-size per parameter type $(n_\sigma = n_\varsigma = n_p = 1)$, instead of n step-sizes $(n_\sigma = n_r, n_\varsigma = n_z, n_p = n_d)$ for all of the variables. This corresponds to the findings in [7].

We also compared single step-size (4,28) MIES to a standard single step-size (4,28) ES on the barrier functions with $C = 20$ and 100 by truncating the continuous parameters to obtain discretized values. The results displayed in Fig. 6 show that the (4,28) MIES perform better than a standard (4,28) ES for the multi-modal barrier function.

5 Application to IVUS Image Analysis

After testing different strategies of MIES on artificial problems, we used MIES to find optimal parameter settings for the segmentation of the lumen in IVUS images. We focused on the lumen detector as it can produce good results in isolation without needing additional information about other IVUS image features.

The settings used for the MIES algorithm were $(\mu = 4, \lambda = 28)$. These parameter settings were found by applying MIES to artificial test problems [16], which were designed to simulate the image problem as much as possible (similar dimension, various levels of difficulty). This was done because evaluating image processing pipelines is very time consuming. The evaluation, for example, of one setting of the MIES algorithm on 40 IVUS images for 100 iterations with 4 parents and 28 offspring took about 16 h on a Pentium 4 (3.4 GHz) computer.

Table 1 lists the parameters for the IVUS lumen image processing pipeline together with their type, range, dependencies and default settings as determined by a human expert. The table shows that the parameters consist of a mix of

[3] Mean value of 20 repeated runs.

Table 1 Parameters for the IVUS lumen image processing pipeline

Name	Type	Range	Dependencies	Default
maxgrey	integer	[2, 150]	$>$ mingrey	35
mingrey	integer	[1, 149]	$<$ maxgrey	1
connectivity	nominal	{4,6,8}		6
relativeopenings	boolean	{false,true}		true
nrofcloses	integer	[0, 100]	used if not relativeopenings	5
nrofopenings	integer	[0, 100]	used if not relativeopenings	45
scanlinedir	nominal	{0,1,2}		1
scanindexleft	integer	[-100, 100]	$<$ scanindexright	-55
scanindexright	integer	[-100, 100]	$>$ scanindexleft	7
centermethod	nominal	{0,1}		1
fitmodel	nominal	{ellipse, circle}		ellipse
sigma	continuous	[0.5 10.0]		0.8
scantype	nominal	{0,1,2}		0
sidestep	integer	[0, 20]		3
sidecost	continuous	[0.0, 100]		5
nroflines	integer	[32, 256]		128

continuous, integer and nominal discrete (including boolean) values, with 16 parameters in total.

The fitness evaluation determines which offspring will serve as new parents in the next generation step. Therefore, the definition of the fitness function is crucially important for a successful application of MIES and should represent the success of the image segmentation very well. We first experimented with a similarity measure $S(c_1,c_2)$ between the contour c_1 found by the lumen feature detector and the desired lumen contour c_2 drawn by a human expert. The similarity measure is defined as the percentage of points of contour c_1 that are less than a $\mathscr{T}$ distance away from contour c_2:

$$S(c_1,c_2) = \frac{\sum_{i=1}^{\text{nrofpoints}} \theta(i)}{\text{nrofpoints}}, \text{with } \theta(i) = \begin{cases} 1 & \text{iff } d(c_1(i),c_2) < \mathscr{T} \\ 0, & \text{otherwise} \end{cases} \tag{6}$$

where $d(c_1(i),c_2)$ is the Euclidian distance between a point i on contour c_1 and contour c_2, nrofpoints is the number of points on contour c_1 and $\mathscr{T}$ is a preset threshold value. This threshold $\mathscr{T}$ determines that two contours are to be considered similar when the distance between all points on contour c_1 is within a distance $\mathscr{T}$ of c_2. The reason a small difference between the two contours is allowed is that even an expert will not draw the exact same contour twice in a single IVUS image. The fitness function itself is the calculated average dissimilarity over all images in a training set.

Although this measure seemed to give good results while looking at the fitness values, visual inspection showed unexpected behaviour, as shown in Fig. 7. The reason for this behaviour is that there was no penalty on the *amount* of distance of contour points from the target contour. As a result, contours with relatively few error

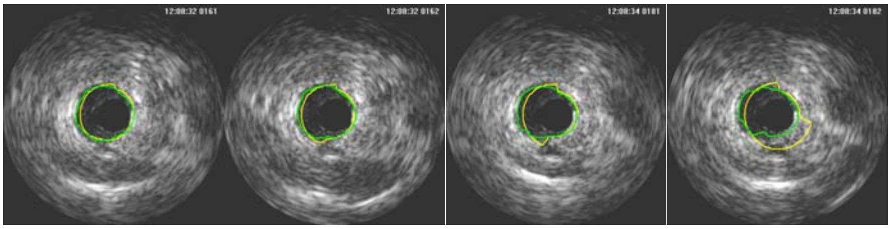


Fig. 7 Expert-drawn lumen contours (*gray*) compared with a MIES parameter solution (*white*) using the similarity measure $S(c_1,c_2)$ (Eq. 6). The images show that large errors may occur even though the fitness of the solution was very good

points could still have a high similarity $S(c_1,c_2)$ value, although visual inspection showed otherwise. To take such effects into account, we changed from maximizing the similarity measure $S(c_1,c_2)$ to minimizing a *dissimilarity* measure $D(c_1,c_2)$, which is defined as follows:

$$D(c_1,c_2) = \sum_{i=1}^{\text{nrofpoints}} \theta(i), \text{ with } \theta(i) = \begin{cases} d(c_1(i),c_2) & \textit{iff } \; d(c_1(i),c_2) > \mathscr{T} \\ 0 & \text{, otherwise} \end{cases} \tag{7}$$

This measure penalizes each contour point that is more than a distance $\mathscr{T}$ away from c_2 and is proportional to the distance. It led to much better results, which stresses the importance of choosing an appropriate fitness function in this problem domain.

An interesting question is whether the MIES obtains better numerical results than the canonical ES using truncation of continuous parameters in order to obtain discretized values. This question was discussed in [16], and in Table 2, we summarize the numerical results on the IVUS problem obtained with MIES, ES and the expert chosen parametrization. In two cases, the ES obtained a equally good result than the MIES, and in the other three cases, it performed much worse. This conforms with the results obtained on the artificial test problems. In summary, we conclude that the use of specialized mutation operators for different parameter types, as it is done in the MIES, will likely improve the performance of the strategy.

Table 2 Performance of the best found ES and MIES parameter solutions on 5 different datasets, compared to an expert chosen parameter solution. Average difference and standard deviation w.r.t. expert-drawn contours are given

Image Set	Expert		ES		MIES	
	Mean	SD	Mean	SD	Mean	SD
Dataset 1	395.2	86.2	149.8	46.0	151.3	39.2
Dataset 2	400.2	109.2	183.0	57.3	181.4	58.7
Dataset 3	344.8	66.4	182.3	49.7	165.6	47.2
Dataset 4	483.1	110.6	232.9	62.4	186.4	59.0
Dataset 5	444.2	90.6	324.7	96.0	171.8	54.5

5.1 Experimental Results

For the experiments, we used five disjoint sets of 40 images. The images were acquired with a 20 MHz Endosonics Five64 catheter using motorized pullback (1 mm/s). Image size is 384×384 pixels (8-bit greyscale) with a pixel size of $0.02602\,\text{mm}^2$. For the fitness function, we took the average dissimilarity over all 40 images with $\mathscr{T}$ set to 2.24 pixels and nrofpoints = 128 (see Eq. 7).

For each of the 5 datasets, we performed 25 runs of the $(4,28)$ MIES algorithm using different random seeds, and limited the number of generation cycles to 25, which resulted in 704 fitness evaluations for each run. The optimization results are displayed in Table 3, where MIES solution 1 was trained on dataset 1 by the MIES algorithm, MIES solution 2 was trained on dataset 2, etc.

Table 3 shows that for most cases, the MIES parameter solutions result in lower average contour differences than the expert chosen default parameters. Only parameter solution 3 applied to dataset 5 has a higher average contour difference (444.2 vs. 446.4) than the default parameters. To determine if the best results obtained by the MIES algorithm are also significantly better than the result of the default parameters, a paired two-tailed Student *T*-test was performed on the (40) different measurements for each image dataset and each solution using a 95% confidence interval (p = 0.05). The *T*-tests show that all differences are statistically significant except for the difference between MIES solution 3 applied to dataset 5 and the default parameters, and the difference between MI-ES solution 5 applied to dataset 3 and the default parameters. Although MIES solution 4 performs slightly better on dataset 5 than MIES solution 5 (171.3 vs. 171.8), which was trained on this dataset, this difference is also not statistically significant. Therefore, we can conclude that the MIES solutions are significantly better than the default default parameter solutions in 92% of the cases (23 out of 25) and equal in the other two cases.

In order to learn from the results about advantageous parameter settings, we compared the variable settings of optimized solutions to solutions with an average fitness value (obtained at the beginning of the evolution). The results are displayed in the parallel coordinates diagram (Fig. 8). It is apparent that the setting of some parameters seems to be clearly advantageous in a certain small range, while for others, the

Table 3 Performance of the best found MIES parameter solutions when trained on one of the five datasets (MIES solution *i* was trained on dataset *i*). All parameter solutions and the expert chosen default parameters are applied to all datasets. Average difference (fitness) and standard deviation w.r.t. expert-drawn contours are given with $\mathscr{T}$ = 2.24 and nrofpoints = 128

	Dataset 1		Dataset 2		Dataset 3		Dataset 4		Dataset 5	
	Mean	SD	Mean	SD	Mean	SD	Mean	SD	Mean	SD
Default	395.2	86.2	400.2	109.2	344.8	66.4	483.1	110.6	444.2	90.6
MIES 1	151.3	39.2	183.6	59.0	201.0	67.1	280.9	91.9	365.5	105.9
MIES 2	160.3	45.9	181.4	58.7	206.7	70.3	273.6	74.5	372.5	99.2
MIES 3	173.8	42.1	202.9	69.1	165.6	47.2	250.7	80.2	446.4	372.8
MIES 4	154.0	51.7	243.7	67.7	198.8	80.1	186.4	59.0	171.3	57.8
MIES 5	275.7	75.6	358.4	76.9	327.7	56.7	329.1	82.0	171.8	54.5

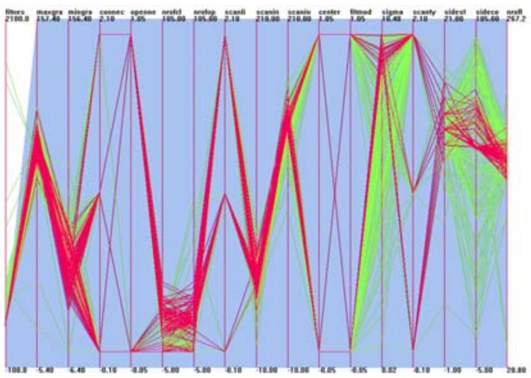


Fig. 8 The parallel coordinate diagrams show a comparison of optimized solutions (*red polygons*) to solutions with a relative bad fitness value (*light green polygons*). The first coordinate is the fitness value and the subsequent coordinates are the values of the variables (Lumen detector parameters) in the order of Table 1

setting is either indifferent or it may depend on other parameter settings. Some observations are that a scantype of 2, a medium value for the maxgrey parameter and sidecost (around 70) and a high sigma value (close to 10.0) seems to be beneficial. Of course, these results hold only for one image set and future research needs to clarify whether generalizations are feasible.

Visual inspection of MIES parameter solution 4 to all 200 images shows that this solution is a good *approximator* of the lumen contours as can be seen in Fig. 9 (bottom row). When we compare the contours generated with MIES solution 4 to the expert-drawn contours, we see that they are very similar, and in some cases, the

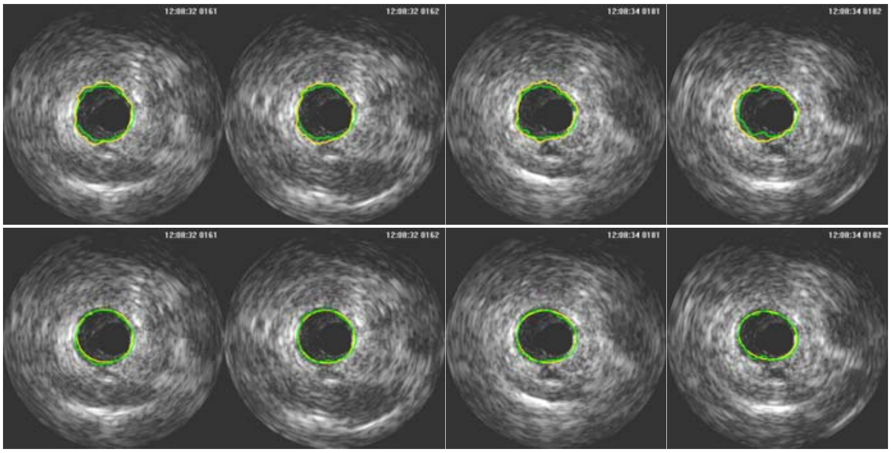


Fig. 9 Expert-drawn lumen contours (*gray*) compared with default parameter solution (*white*, top row) and MIES parameter solution (*white*, bottom row)

MIES contours actually seem to follow the lumen boundary more precisely. Besides being closer to the expert-drawn contours, another major difference between the MIES found contours and the contours detected with the default parameter settings is that the MIES solutions produce more smooth contours (see Fig. 9, top and bottom rows).

6 Conclusions and Future Work

In this chapter, we described the MIES and applied it to a problem in medical image analysis, particularly the optimization of control parameters of a lumen detector in IVUS imaging.

MIES uses specific variation operators for different types of decision variables (continuous, integer and nominal discrete). All three types of mutation operators support automatic adaptation of the mutation strength and avoid biased sampling. In addition, they fulfill guidelines such as accessibility, uniformity and maximal entropy, which makes them very amenable as search operators in settings with little or no *a priori* knowledge about the search landscape. To determine favourable default settings, we tested the MIES on artificial test problems. Moreover, we compared MIES to the standard (continuous) ES using simple truncation of continuous variables. It turns out that the MIES approach has a much higher convergence reliability.

A similar result is obtained for the medical image analysis. Here, the MIES always produced better or equal results than the default parameter settings chosen by an expert. Moreover, on all five data sets, the results of the MIES were significantly better (three times) or equal (one time) than those obtained with the standard ES, trained on the same data set.

In summary, the results show that the MIES is a valuable technique for improving the parameter settings of the lumen detector. The results encourage further studies on extended image sets and for other feature detectors. The results of this study also suggest its use in other problems where parameters of image analysis modules need to be tuned based on training data, and, more generally, for large-scale mixed-integer optimization problems that cannot be solved with standard mathematical programming techniques.

With regard to the image analysis problem, we do not expect to find one optimal solution for each feature detector to work in all possible contexts and for all possible patients. Therefore, we would like to let a set of MIES algorithms find a set of optimal solutions for sets of optimal images, whereby the solutions and sets of images are evolved automatically. A first approach towards this fitness-based partitioning is described in [20, 21]. After we have evolved the set of optimal image feature detector solutions, the aim is then to let an agent in the multiagent system decide which particular solution to use based on its current knowledge of the situation.

We also intend to further study the MIES algorithm both in theory and practice. In particular, approaches that enable us to integrate more problem knowledge, such as exploiting dependencies between different variables, are promising future extensions.

Acknowledgement. This research was supported by the Netherlands Organization for Scientific Research (NWO) and the Technology Foundation STW.

References

1. Bovenkamp, E., Dijkstra, J., Bosch, J., Reiber, J.: Multi-agent segmentation of ivus images. Pattern Recognition 37(4), 647–663 (2004)
2. Papadogiorgaki, M., Mezaris, V., Chatzizisis, Y.S., Giannoglou, G.D., Kompatsiaris, I.: Automated ivus contour detection using intesity features and radial basis function approximation. In: CBMS 2007: Proceedings of the Twentieth IEEE International Symposium on Computer-Based Medical Systems, pp. 183–188. IEEE Computer Society, Washington (2007)
3. Sanz-Requenaa, R., Moratala, D., García-Sáncheza, D.R., Bodíb, V., Rietaa, J., Sanchis, J.: Automatic segmentation and 3d reconstruction of intravascular ultrasound images for a fast preliminar evaluation of vessel pathologies. Computerized Medical Imaging and Graphics 31(2), 71–80 (2007)
4. Bovenkamp, E., Dijkstra, J., Bosch, J., Reiber, J.: User-agent cooperation in multi-agent ivus image segmentation. IEEE Transactions on Medical Imaging (accepted) (2008)
5. Newell, A.: Unified Theories of Cognition. Harvard University Press, Cambridge (1990)
6. Ballerini, L., Franzén, L.: Genetic optimization of morphologicial filters with applications in breast cancer detection. In: Raidl, G.R., Cagnoni, S., Branke, J., Corne, D.W., Drechsler, R., Jin, Y., Johnson, C.G., Machado, P., Marchiori, E., Rothlauf, F., Smith, G.D., Squillero, G. (eds.) EvoWorkshops 2004. LNCS, vol. 3005, pp. 250–259. Springer, Heidelberg (2004)
7. Emmerich, M., Grötzner, M., Groß, B., Schütz, M.: Mixed-integer evolution strategy for chemical plant optimization with simulators. In: Parmee, I. (ed.) Evolutionary Design and Manufacture - Selected papers from ACDM, pp. 55–67. Springer, London (2000)
8. Rosamond, W., Flegal, K., Friday, G., Furie, K., Go, A., Greenlund, K., Haase, N., Ho, M., Howard, V., Kissela, B., Kittner, S., Lloyd-Jones, D., McDermott, M., Meigs, J., Moy, C., Nichol, G., O'Donnell, C.J., Roger, V., Rumsfeld, J., Sorlie, P., Steinberger, J., Thom, T., Wasserthiel-Smoller, S., Hong, Y.: Heart disease and stroke statistics–2007 update: a report from the american heart association statistics committee and stroke statistics subcommittee. Circulation 115(5) (2007)
9. Bäck, T.: Evolutionary Algorithms in Theory and Practice. Oxford University Press, New York (1996)
10. Rechenberg, I.: Evolutionsstrategie 1994. Frommann-Holzboog, Stuttgart (1994)
11. Schwefel, H.P.: Evolution and Optimum Seeking. Sixth-Generation Computer Technology Series. Wiley, New York (1995)
12. Rudolph, G.: An evolutionary algorithm for integer programming. In: Davidor, Y., Männer, R., Schwefel, H.-P. (eds.) PPSN 1994. LNCS, vol. 866, pp. 139–148. Springer, Heidelberg (1994)
13. Schütz, M., Sprave, J.: Application of parallel mixed-integer evolution strategies with mutation rate pooling. Evolutionary Programming, 345–354 (1996)

14. Groß, B.: Gesamtoptimierung verfahrenstechnischer Systeme mit Evolutionären Algorithmen. Dissertation, Rheinisch – Westfälische Technische Hochschule Aachen, Lehrstuhl für Technische Thermodynamik (1999)
15. Li, R., Emmerich, M., Bovenkamp, E., Eggermont, J., Bäck, T., Dijkstra, J., Reiber, J.: Mixed-Integer Evolution Strategies and Their Application to Intravascular Ultrasound Image Analysis. In: Rothlauf, F., Branke, J., Cagnoni, S., Costa, E., Cotta, C., Drechsler, R., Lutton, E., Machado, P., Moore, J.H., Romero, J., Smith, G.D., Squillero, G., Takagi, H. (eds.) EvoWorkshops 2006. LNCS, vol. 3907, pp. 415–426. Springer, Heidelberg (2006)
16. Li, R., Emmerich, M., Eggermont, J., Bovenkamp, E., Bäck, T., Dijkstra, J., Reiber, J.: Mixed-Integer Optimization of Coronary Vessel Image Analysis using Evolution Strategies. In: Keijzer, et al. (eds.) GECCO 2006: Proceedings of the 8th annual conference on Genetic and evolutionary computation, Seattle, WA, USA, pp. 1645–1652 (2006)
17. Hoffmeister, F., Sprave, J.: Problem-independent handling of constraints by use of metric penalty functions. In: Fogel, L., et al. (eds.) Evolutionary Programming 1996, San Diego, CA, USA, NY, USA, pp. 289–294. IEEE Press, Los Alamitos (1996)
18. Beyer, H.G.: Theory of Evolution Strategies. Springer, Heidelberg (2001)
19. Rudolph, G.: Convergence Properties of Evolutionary Algorithms. Schriftenreihe Forschungsergebnisse zur Informatik. Kovac (1997)
20. Eggermont, J., Li, R., Bovenkamp, E.P., Marquering, H., Emmerich, M.M., van der Lugt, A., Bäck, T., Dijkstra, J., Reiber, J.C.: Optimizing computed tomographic angiography image segmentation using fitness based partitioning. In: Giacobini, M., Brabazon, A., Cagnoni, S., Di Caro, G.A., Drechsler, R., Ekárt, A., Esparcia-Alcázar, A.I., Farooq, M., Fink, A., McCormack, J., O'Neill, M., Romero, J., Rothlauf, F., Squillero, G., Uyar, A.Ş., Yang, S. (eds.) EvoWorkshops 2008. LNCS, vol. 4974, pp. 275–284. Springer, Heidelberg (2008)
21. Li, R., Eggermont, J., Emmerich, M.M., Bovenkamp, E., Bäck, T., Dijkstra, J., Reiber, J.C.: Towards dynamic fitness based partitioning for intravascular ultrasound image analysis. In: Giacobini, M. (ed.) EvoWorkshops 2007. LNCS, vol. 4448, pp. 391–398. Springer, Heidelberg (2007)

Memetic Differential Evolution Frameworks in Filter Design for Defect Detection in Paper Production

Ferrante Neri and Ville Tirronen

Abstract. This chapter studies and analyzes Memetic Differential Evolution (MDE) Frameworks for designing digital filters, which aim at detecting paper defects produced during an industrial process. MDE Frameworks employ the Differential Evolution (DE) as an evolutionary framework and a list of local searchers adaptively coordinated by a control scheme. Here, three different variants of MDE are taken into account and their features and performance are compared. The binomial explorative features of the DE framework in contraposition to the exploitative features of the local searcher are analyzed in detail in light of the stagnation prevention problem, typical for the DE. Much emphasis in this chapter is given to the various adaptation systems and to their applicability this image processing problem.

1 Introduction

In recent years, machine vision systems for quality inspection and fault detection have become standard in paper industry. These systems monitor the paper web for structural defects such as holes and defects in quality such as faint streaks, thin spots and wrinkles [1]. Detection of weak defects is crucial in quality paper production since their presence in paper sheets raises difficulties, for example, in printing, thus leading to significant monetary losses. The paper web inspection is a challenging task since it must be executed under strict real-time constraints. In fact, paper machines can achieve a speed of over 25 m/s while the defects are barely a few millimetres in size.

Defect detection falls within the spectrum of low-level vision since it is related to both edge detection and textural analysis and can be seen as a pre-segmentation technique for identifying defects and insulating them from the background.

In order to solve this class of problems, several solutions have been proposed over the years in literature and industrial applications. Classical edge detection methods

Ferrante Neri and Ville Tirronen
Department of Mathematical Information Technology, Agora, University of Jyväskylä, P.O. Box 35 (Agora), FI-40014 University of Jyväskylä, Finland
e-mail: neferran@cc.jyu.fi,aleator@cc.jyu.fi

and simple segmentation schemes such as plain threshold have been widely studied in literature [2] and have become the toolbox of machine vision engineering. These methods, in spite of their popularity with the industry, are often fragile and need to be tuned by human experts for each new condition.

Thus, Computational Intelligence (CI) approaches have been successfully applied in low-level vision during recent years. For example, such approaches aim at executing machine learning by means of neural networks [3, 4] or employing evolutionary algorithms to derive high-performance operators for specific applications [5–8]. The latter approach is very promising because these operators are potentially adaptable in different situations without human expert decision making [9].

This chapter aims at studying the defect detection in paper production by means of image processing techniques. In our application, the high real-time requirements pose a serious limitation to applicable techniques. In addition, the defects studied in this application are rather faint and masked by the natural structure of the paper, which varies with time and process state. Moreover, compared to other fields of defect detection, we lack regular texture or estimable background, as encountered with, for example, textiles [10].

Defect detection in paper production has been studied over the years and several solutions have been proposed in industries and literature. The most commonly used approaches in industrial applications are based on simple thresholding techniques [11]. Since these techniques are inadequate for weak defects, more sophisticated methods are required. A popular approach is to utilize texture-based techniques. In [12], the defects are characterized as deviations from the background texture; the problem is encoded as a two-class classification problem by means of local binary patterns as a source of features and a Self-Organizing Map (SOM) as a clustering/classification tool. However, wrinkles and streaks are often quite faint perturbations of texture, and can be easily overlooked with texture analysis.

This work proposes an approach which employs Finite Impulse Response (FIR) filters, parameterized by a Gabor function, specifically optimized for the task of weak defect detection. The problem-oriented design of the FIR filters seems very promising for similar image processing problems in other fields of application, for example, in [13] for texture segmentation and in [14] for a vehicle tracking problem. Sun et al. [14] also proposed a hybrid evolutionary algorithm in order to perform the filter design. FIR filters are flexible and have relatively efficient machine implementations, which makes them applicable for the task. Gabor functions are useful for filter formulation in cutting down the number of parameters and are presumed applicable due to similarities with mammalian vision cortex responses.

2 Features of the Filter and Problem Formulation

Training of the filter is based on a set of source images supplied by Viconsys Oy[1] and a corresponding set of label images. These two sets constitute the training set. Images are taken in a static situation with a resolution of approximately

[1] `http://www.viconsys.fi/`

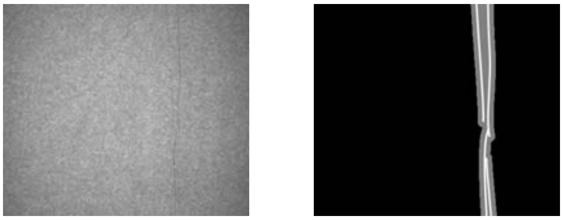


Fig. 1 Source and Label image

(a) Source image (b) Label image

0.6 mm/pixel. Figure 1a shows an image belonging to the training set with the corresponding label image (Fig. 1b).

We propose the employment of Gabor filters [15] since they turned out to be very successful in similar applications of machine vision for defect detection [16, 17] and edge detection [18]. In particular, the following Gabor function [15] is proposed:

$$Gb[\theta,\psi,\sigma_x,\sigma_y,\lambda](x,y) = e^{\frac{-(x\cos\theta - y\sin\theta)^2}{2\sigma_x^2}} \cdot e^{\frac{-(x\sin\theta + y\cos\theta)^2}{2\sigma_y^2}} \cdot \cos\left(\frac{2\pi(x\cos\theta - y\sin\theta)}{\lambda} + \psi\right), \tag{1}$$

where θ is the angle between the direction perpendicular to parallel stripes of the Gabor function (the plot of a two-dimensional Gabor function always generates a set of parallel stripes; see [15]) and a vertical reference axis; ψ is the phase offset (the filter is symmetrical when $\psi = 0$ and antisymmetrical when $\psi = \frac{\pi}{2}$). Furthermore, σ_x, σ_y specify both the size of the filter and it's ellipticity, and λ is the wavelength of the filter. In other words, we use a directed bandpass filter with bandwidth determined by λ and the ratio $\frac{\sigma_x}{\sigma_y}$.

Gabor filters are limited by their relatively high computational cost. The real-time requirements of our application rule out the use of Fast Fourier Transform (FFT) and computation of filters via convolution theorem. To satisfy real-time requirements in the spatial domain, we sample the Gabor function into a 7×7 discrete kernel. This will limit the available scales and frequencies. However, since the problem is empirical, we allow the filter to be truncated to kernel size, trading accuracy to overcome the aforementioned limits [19].

Since the defects tend to contain large variations in angle and the filter is direction dependent, the outputs of two different filters are combined.

The coordination of the filters is carried out, for each pixel, by multiplying filter output by a weight coefficient and then selecting the output (i.e. the filter), which produces maximal intensity (see formula (2)). A post-filtering stage, by means of a gaussian filter, is included in order to mitigate the effect of spurious responses due to noise [20]. Thus, for a given image I, α indicating the vector of 12 elements representing the design parameters of the two filters, the filter formula is given by

$$\begin{aligned} F(\alpha, I) = G_{15,15} \star \max\ &[(\alpha(1)\, Gb(\alpha(2), \ldots, \alpha(6)) \star I), \\ &(\alpha(7)\, Gb(\alpha(8), \ldots, \alpha(12)) \star I)], \end{aligned} \tag{2}$$

where $\alpha(1)$ and $\alpha(7)$ are the weights for both filters, $\star$ denotes the two-dimensional discrete convolution and $G_{r,t}$ represents an ordinary gaussian kernel of size $r \times t$. The design parameters of the Gabor filters are shown in Table 1.

The problem of the filter design thus consists of finding a proper set of design parameters α. In order to estimate the fitness of a candidate solution α, the following procedure is carried out. Let us indicate the source image with S and the corresponding label image with L.

The filtered images F are divided into three regions based on the label images: the defect region D is defined as the set of those pixels (x,y) of the image F such that $L(x,y) = 1$, the clean region C is defined as the set of those pixels (x,y) such that $L(x,y) = 0$ and the non-interesting region is characterized by other colours (grey in Fig. 1b). Let us define the similarity function sim as follows:

$$\mathrm{sim}(S,L) = a\sigma(D) + b\sigma(C) + c\sqrt{|\mu(D) - \mu(C)|}, \tag{3}$$

where μ and σ denote mean and standard deviation over the set of pixels and a, b and c are weight coefficients. The first term in (3) measures uniformity of the defect region, the second term measures background noise (C region) and the third term measures separation of the two regions. The weight coefficients have been set as $a = 1$, $b = 2$ and $c = -3$ taking into consideration that it is highly desirable for the filter to clearly separate defects from the background, and it is also important that noise in the background does not lead to detection of false defects. Finally, the fitness function f is given by

$$f(\alpha) = \frac{1}{n_I} \sum_{k=1}^{n_I} \mathrm{sim}\left(F(\alpha, S_k), L_k\right), \tag{4}$$

where n_I is the total number of images in the training set and S_k and L_k are the kth source and label image from the training set, respectively. The filter design is thus stated as the minimization of f over $H = [-100, 100]^2 \times [-\pi, \pi]^2 \times [0, \pi]^2 \times [1, 20]^6$.

Table 1 Design parameters

Parameter	Description	Range of variability
$\alpha(1), \alpha(7)$	Weight	$[-100, 100]$
$\alpha(2), \alpha(8)$	θ	$[-\pi, \pi]$
$\alpha(3), \alpha(9)$	ψ	$[0, \pi]$
$\alpha(4), \alpha(10)$	σ_x	$[1, 20]$
$\alpha(5), \alpha(11)$	σ_y	$[1, 20]$
$\alpha(6), \alpha(12)$	λ	$[1, 20]$

3 Differential Evolution and Prior Hybridizations

Differential Evolution (DE) [21] is a reliable and versatile function optimizer. DE, like most popular Evolutionary Algorithms (EAs), is a population-based tool. DE, unlike other EAs, generates offspring by perturbing the solutions with a scaled difference of two randomly selected population vectors, instead of recombining the solutions by means of a probability function. In addition, DE employs a steady-state logic which allows replacement of an individual only if the offspring outperforms its corresponding parent.

Due to its algorithmic structure, over the optimization process, DE generates a super-fit individual which leads the search until an individual with better performance is generated. As highlighted in [22], a DE population can be subjected to stagnation in the case where no individuals outperform the corresponding parents for a large number of generations. On the other hand, the persistence of a super-fit individual over a certain number of generations does not necessarily imply poor functioning of the algorithm; this may be a “natural” stage of evolution (i.e. a proper functioning of the DE) if the other individuals are somehow updated during the run. It is therefore fundamental to find a proper balance between the enhancements of the search leader (super-fit individual) and the remaining individuals of the population.

In order to prevent stagnation, several studies have been carried out in recent years. In [22], based on an experimental study, some suggestions on how to perform the parameter setting are given. In [23], a dynamic parameter setting that changes with time is proposed. A further study on the proper parameter setting in a DE framework is performed in [24], and a fuzzy logic-based criterion in order to automatically solve this problem is proposed in [25]. This fuzzy logic approach has been analyzed in depth in [26] and [27]. Ali and Törn [28] propose a fitness-based adaptive setting of the scale factor. Tvrdik [29] extends the work in [28] and proposes the concept of competition during on-line parameter setting. In [30], randomizing the scale factor of the perturbation vector is proposed. This operation leads to an increase in the pool of potential trial vectors, thus reducing the risk of stagnation without increasing the population size. Recently, a further study on the DE parameter setting, with reference to real-world problems, has been carried out in [31].

In order to enhance performance, several approaches which hybridize the DE with other optimization techniques have also been considered. In [32], the DE framework is embedded with two additional operators, which increase the convergence velocity of the DE without jeopardizing the algorithmic features in terms of population diversity. In [33], an enhancement of the algorithm in [32] for solving constrained optimization problems is proposed. Mydur [34] proposes a hybridization of the DE with the Powell method for accelerating the DE performance with reference to an electrical engineering problem. Other hybridizations with single local search algorithm are given in [35] and [36]. In [37], the combination of the DE with Particle Swarm Optimization Algorithm is shown. Lin et al. [38] propose a hybrid DE for mixed integer-real coded problems. Lin et al. and Su and Lee [39] and [40] propose a co-evolutionary and an enhanced implementation of the algorithm described in [38], respectively. In [41], the DE has been hybridized with a multiplier updating method for constrained problems. Chiou et al. [42] proposes the hybridization of the DE with an Ant Colony Optimization algorithm and [43] with the Salomon's Evolutionary Gradient Search method. In [44], a hybrid DE with a mutation local searcher based on the attract–repulsion logic is proposed.

4 Memetic Differential Evolution Frameworks

This section describes three memetic algorithms designed in order to perform the minimization of f in H shown in Sect. 2. These algorithms employ DE as an evolutionary framework and a list of local searchers coordinated by an adaptive system. The first algorithm, namely MDE [45], makes use of the Hooke–Jeeves Algorithm (HJA) [46] and a Stochastic Local Searcher (SLS) [47] adaptively coordinated by a fitness diversity-based index. The second, namely Enhanced Memetic Differential Evolution (EMDE), employs the HJA, a Stochastic Local Searcher and Simulated Annealing (SA) [48] coordinated by means of a fitness diversity adaptation and a probabilistic scheme. The third, namely Super-Fit Memetic Differential Evolution (SFMDE) [49], employs Particle Swarm Optimization (PSO), the Nelder–Mead Algorithm (NMA) and the Rosenbrock Algorithm (RA). The coordination of the local searchers is performed by means of an index measuring the quality of the super-fit individual with respect to the rest of the population and a probabilistic scheme based upon the generalized beta distribution.

4.1 The Memetic Differential Evolution

An initial sampling of S_{pop} individuals is executed pseudo-randomly with a uniform distribution function over the decision space H. At each generation, for S_{pop} times, four individuals, α_1, α_2, α_3 and α_4, are extracted from the population pseudo-randomly. Recombination according to the logic of a DE occurs at first by generating α'_{off} according to the following formula [21, 50]:

$$\alpha'_{\text{off}} = \alpha_1 + K(\alpha_2 - \alpha_3), \tag{5}$$

where $K = 0.7$ is a constant value set according to the suggestions given in [21]. Then, in order to increase the exploration of this operator, a random component is introduced by switching some design parameters of α'_{off} with the corresponding genes of α_4. Each switch occurs with a uniform mutation probability $p_{\text{m}} = 0.3$, as suggested in [22], and the offspring α_{off} is thus generated. The fitness value of α_{off} is calculated and, according to a steady-state strategy, if α_{off} outperforms α_4, it replaces α_4; if, on the contrary, $f(\alpha_{\text{off}}) > f(\alpha_4)$, no replacement occurs. The MDE employs the following two local searchers which assist the DE framework by offering alternative exploratory perspectives.

The HJA [46] initializes the exploratory radius $h_{\text{HJA}-0}$, an initial candidate solution α and a 12×12 direction exploratory matrix $U = \text{diag}(w(1), w(2) \dots w(12))$, where $w(m)$ is the width of the range of variability of the mth variable. Let us indicate with $U(m,:)$ the mth row of the direction matrix $m = 1,2\dots 2$. The HJA consists of an exploratory move and a pattern move. Indicating with h_{HJA} the current best candidate solution with α and the generic radius of the search, the HJA during the exploratory move samples points $\alpha(m) + h_{\mathrm{HJA}}U(m,:)$ ("+" move) with $m = 1,2\dots 12$ and the points $\alpha(m) - h_{\mathrm{HJA}}U(m,:)$ ("-" move) with $m = 1,2\dots 12$ only along those directions which turned out unsuccessful during the "+" move. If a new current best is found, α is then updated and the pattern move is executed. If a new current best is not found, h_{HJA} is halved and the exploration is repeated. The HJA pattern move is an aggressive attempt of the algorithm to exploit promising search directions. Rather than centering the following exploration at the most promising explored candidate solution (α), the HJA tries to move further. The algorithm centres the subsequent exploratory move at $\alpha \pm h_{\mathrm{HJA}}U(m,:)$ ("+" or "-" on the basis of the best direction). If this second exploratory move does not outperform $f(\alpha)$ (the exploratory move fails), then an exploratory move with α as the centre is performed. The HJA stops after 1000 fitness evaluations (budget condition).

The SLS [47] picks up a solution α and initializes a parameter $\sigma_{\mathrm{SLS}-0}$. Then, 24 perturbation vectors h_{SLS} are generated; these vectors have the same length of α and each gene is a random number of a normal distribution having a mean value of $\alpha(m)$ and standard deviation $\sigma_{\mathrm{SLS}}w(m)$. Number of perturbations is chosen so that SLS and HJA have similar computation costs. For each of these perturbation vectors, $\alpha + h_{\mathrm{SLS}}$ is calculated and the related fitness value is saved. If the most successful perturbation has better performance than the starting solution, the perturbed solution replaces the starting one; otherwise, σ_{SLS} is halved and the process is repeated. The algorithm stops when the budget on the number of fitness evaluations (1000) is exceeded.

In order to perform coordination of the local searchers, the MDE uses a novel adaptive scheme. For every 1000 fitness evaluations, the following index is calculated:

$$v = \min\left\{1, \frac{\sigma_{\mathrm{f}}}{\left|f_{\mathrm{avg}}\right|}\right\}, \tag{6}$$

where $\left|f_{\mathrm{avg}}\right|$ and σ_{f} are, respectively, the average value and standard deviation over the fitness values of individuals of the population. The parameter v can vary between 0 and 1 and can be seen as a measurement of fitness diversity and distribution of the fitness values within the population [51, 52]. More specifically, if $v \approx 0$, the fitness values are similar to each other; on the contrary, if $v \approx 1$, the fitness values are different to each other and some individuals thus perform much better than the others [53, 54]. This index is used to activate local searchers according to the following scheme:

(a) if $v \geq 0.5$, no local searchers are activated and the DE is continued for 1000 fitness evaluations;
(b) if $0.4 \leq v < 0.5$, one individual of the population is pseudo-randomly chosen and the SLS is applied to it;

(c) if $0.25 \leq \nu < 0.4$, the SLS is applied to an individual pseudo-randomly chosen and the HJA is applied to the individual that has the best performance and
(d) if $\nu < 0.25$, the HJA is applied to the individual which has the best performance

According to our algorithmic philosophy, when ν is small, local searchers are activated in order to offer a different explorative perspective to the DE framework and hopefully detect new promising genotypes. On the other hand, for high values of ν, the DE framework is supposed to exploit the highly diverse population genotypes by recombination. Since the two local searchers have different structures and features, they are activated by means of different threshold values of ν. More specifically, when $0.25 \leq \nu < 0.5$, the DE framework reduced the fitness diversity of the population, but still has some genotypes that can likely be exploited. Under this condition, the SLS is supposed to assist the DE attempting to improve available genotypes that can be used by the DE crossover. On the contrary, when ν is very small (e.g. $\nu = 0.2$), fitness diversity is low and thus the end of the optimization process is approaching, meaning that the solutions are either all contained in the same basin of attraction or spread out in different areas of the decision space with similar performance [53, 54]. In this case, the SLS is no longer efficient [55]. The HJA is therefore activated in order to execute a deterministic hill-descent of the basin of attraction from the best performing solution with the aim to either include, in the population, a new genotype having high performance or possibly end the game of the optimization process.

Finally, ν is also used to initialize the parameters $\sigma_{SLS-0} = \nu/2$ and $h_{HJA-0} = \nu/2$. This choice means that the initial radius of exploration of the local searchers should be large when fitness diversity is high and small when fitness diversity is low. More specifically, the smaller ν, the closer the end of the optimization process (see [53–55]); when ν is small, the local searchers attempt to detect a better performing solution within the neighbourhood of the starting solution and when ν is large, the local searchers have a more explorative behaviour. The algorithm stops when a budget condition is exceeded. The MDE pseudo-code is shown in Fig. 2.

```
generate initial population pseudo-randomly;
while budget condition
    for 1000 fitness evaluations
        execute DE recombination and offspring generation;
    end-for
    compute $\nu = \min\left\{1, \frac{\sigma_f}{|f_{avg}|}\right\}$;
    if $0.25 \leq \nu < 0.5$
        execute SLS on an individual pseudo-randomly selected;
    end-if
    if $0.2 \leq \nu < 0.4$
        execute HJA on the individual having the best performance;
    end-if
end-while
```

Fig. 2 MDE pseudo-code

4.2 The Enhanced Memetic Differential Evolution

The EMDE employs the DE framework, the HJA and SLS, as described in Sect. 4.1. The EMDE also employs the SA metaheuristic [48, 56] in order to offer a third exploratory perspective in the decision space. In particular, the SA has been selected since it can choose a search direction leading to a basin of attraction different from where the starting candidate solution α_0 is. The exploration is performed by using the same perturbation logic as described in the SLS (see Sect. 4.1). The temperature $Temp$ is reduced according to a hyperbolic law following the suggestions in [57]. Unlike the case of the MDE, all the local searchers are run each time for 500 fitness evaluations (budget condition). This algorithmic choice has been carried out due to the empirical observation that most improvements of the local search were obtained in the early runs.

In order to perform coordination of the local searchers, an adaptive functioning is proposed. For every 1000 DE fitness evaluations, the index ν shown in (6) is calculated. A value ε is sampled by means of a uniform distribution function within the interval $[0,1]$. Then, the following exponential distribution function is considered [58]:

$$P(\mu_p, \sigma_p, \nu) = e^{\frac{-(\nu-\mu_p)}{2\sigma_p^2}} \tag{7}$$

where $\sigma_p = 0.1$ and $\mu_p \in \{0.1, 0.2, 0.3, 0.4\}$. The following probabilistic scheme is proposed.

(a) if $\varepsilon < P(0.4, \sigma_p, \nu)$, one individual is pseudo-randomly extracted from the population and the SA is applied for 500 fitness evaluations;
(b) if $\varepsilon < P(0.3, \sigma_p, \nu)$, the HJA is applied for 500 fitness evaluations to the individual having the best performance;
(c) if $\varepsilon < P(0.2, \sigma_p, \nu)$, one individual is pseudo-randomly extracted from the population and the SLS is applied for 500 fitness evaluations and
(d) if $\varepsilon < P(0.1, \sigma_p, \nu)$, the SLS is applied for 500 fitness evaluations to the individual having the best performance

Conditions (a) and (c) mean that alternative exploratory perspectives are offered to the DE framework and it is taken into account that, for a bad initial solution (early stage of evolution), the SA performs better than the SLS, and conversely, for a good initial solution (late stage of evolution), the SLS performs better than the SA. Conditions (b) and (d) are related to the best individual of the population. Condition (b) states that a promising basin of attraction must be immediately descended by means of a highly exploitative local searcher (HJA). On the contrary, condition (d) is related to the late stages of the evolution and states that finalization of the optimization process can be more efficiently performed by a local searcher rather than an evolutionary framework.

In addition, two heuristic rules have been employed. According to the first rule, if the application of SA does not lead to any improvement, the initial solution is not replaced. According to the second, the HJA is never applied twice to the same individual if a previous HJA application did not already lead to any improvement.

Finally, the initial temperature $Temp_0$ is adaptively set to be $Temp_0 = \nu$. This means that the probability of accepting a worse solution depends on fitness diversity. In other words, the algorithm does not accept worse solutions when the fitness diversity is low and thus the SA does not attempt, with a high probability, to accept solutions that have worse performance than the initial one.

The algorithm stops when a budget condition has been exceeded. Figure 3 shows the EMDE pseudo-code.

Fig. 3 EMDE pseudo-code

```
generate initial population pseudo-randomly;
while budget condition
    for 1000 fitness evaluations
        execute DE recombination and offspring generation;
    end-for
    compute $\nu = \min\left\{1, \frac{\sigma_f}{|favg|}\right\}$;
    sample $\varepsilon \in [0,1]$
    if $\varepsilon < e^{\frac{-(\nu-0.4)}{2\sigma_p^2}}$
        execute SA on an individual pseudo-randomly selected;
    end-if
    if $\varepsilon < e^{\frac{-(\nu-0.3)}{2\sigma_p^2}}$
        execute HJA on the individual having the best performance;
    end-if
    if $\varepsilon < e^{\frac{-(\nu-0.2)}{2\sigma_p^2}}$
        execute SLS on an individual pseudo-randomly selected;
    end-if
    if $\varepsilon < e^{\frac{-(\nu-0.1)}{2\sigma_p^2}}$
        execute SLS on the individual having the best performance;
    end-if
end-while
```

4.3 *Super-Fit Memetic Differential Evolution*

To begin with, the algorithm generates the initial population by pseudo-randomly sampling (uniform distribution) S_{pop} individuals within the decision space. Fitness values of these individuals are calculated and the one having the best performance is detected.

The best performing solution and $S_{\text{pop}}^{\text{PSO}} \leq S_{\text{pop}}$ individuals, pseudo-randomly selected from the initial population, undergo PSO [59, 60]. The best performing solution initializes the particle best α_{pb} (the best solution overall detected), while the best performing solution amongst the $S_{\text{pop}}^{\text{PSO}}$ individuals is called the global best α_{gb}. At each generation, the individuals are perturbed by means of a velocity vector. For a given solution α_i, the update rule is given by $v_i = v_i + c_1\gamma\left(\alpha_{\text{pb}} - \alpha_i\right) + c_2\gamma\left(\alpha_{\text{gb}} - \alpha_i\right)$ and $\alpha_i = \alpha_i + v_i$ where γ is a pseudo-random value sampled in $[0,1]$ and $c_1 = c_2 = 2$ are constant parameters called learning factors. The fitness values of the new solutions are calculated and, if an improvement upon the best individuals occurs, α_{pb} and α_{gb} are also updated. The procedure is repeated as long as the budget condition is not exceeded. The best overall solution detected by the PSO is reinserted in the population made up of S_{pop} individuals by replacing the individual having the worst performance.

The main idea is that the PSO should quickly improve a solution having poor performance and include it in the DE population. This solution should therefore be a super-fit individual, with the role of leading the DE search. According to our algorithmic philosophy, the DE should then exploit the genotypic information of the solution returned by the PSO, and at the same time, attempt to improve upon it by a massive exploration of the decision space.

When the super-fit individual is generated by the PSO, the population is subject to the same DE framework described in Sect. 4.1. The SFMDE also employs the NMA and the RA.

The NMA [61] works on a set of $n+1$ solutions in order to perform a local search since it employs an exploratory logic based on a dynamic construction of a polyhedron (simplex). More specifically, for a given set of $n+1$ solutions, $\alpha_0, \alpha_1, \ldots, \alpha_n$, sorted in descending order according to their fitness values (i.e. x_0 is the best), the NMA attempts to improve α_n. In order to pursue this aim, the NMA calculates the centroid α_m of the polyhedral identified by the remaining n points: $\alpha_m = \frac{1}{n}\sum_{j=0}^{n-1} \alpha_j$. Subsequently, the NMA generates a trial solution α_r by reflection. Reflection is performed by $\alpha_r = \alpha_m + F1(\alpha_m - \alpha_n)$, where $F1$ is a weighting factor. If a reflection is successful i.e. $f(\alpha_r) < f(\alpha_0)$, NMA further exploits a promising search direction by applying $\alpha_e = \alpha_r + F2(\alpha_m - \alpha_n)$, where $F2$ is a weighting factor. If the expansion is also successful, α_e replaces α_0 and the new set of $n+1$ points is used for the subsequent iteration. Conversely, if the expansion fails, then α_r replaces α_0.

If α_r did not improve upon α_0, NMA compares the performance of α_r and α_{n-1}. If $f(\alpha_r) < f(\alpha_{n-1})$, then α_r replaces α_n. If this trial is also unsuccessful, α_r and α_n are compared. If $f(\alpha_r) < f(\alpha_n)$, however, α_r replaces α_n and the outside contraction operation is executed: $\alpha_c = \alpha_m + F3(\alpha_m - \alpha_n)$, where $F3$ is a weighting factor. If α_r does not outperform α_n then the inside contraction is executed: $\alpha_c = \alpha_m - F3(\alpha_m - \alpha_n)$. The effect of the contraction is then analyzed: if the contraction was successful, i.e. $f(\alpha_c) < f(\alpha_n)$, then α_c replaces α_n. If, on the contrary, contraction fails, the shrinking is executed. A set of n solutions $\alpha_j = 0.5(\alpha_0 - \alpha_j)$ with $j = 1, 2, \ldots, n$, is generated around x_0.

The algorithm is repeated with these new n solutions and x_0. $F1$, $F2$ and $F3$ have been set equal to 1, 1 and 0.5, respectively, as suggested in [62].

The RA works on a solution and attempts to improve upon it by means of a local search logic [63]. From a starting point α_0, a trial is made in all the n orthogonal directions of the n-dimensional decision space. When a success is scored (including equality of the objective function values), the changed variable vector is retained and the step length is multiplied by a positive factor $\gamma > 1$. For a failure, the vector of variables is left unchanged and the step length is multiplied by a negative factor $-1 < \beta < 0$. Following the Rosenbrock's suggestion, γ has been set to 3 and β to 0.5. This process is repeated until at least one success followed by a failure is

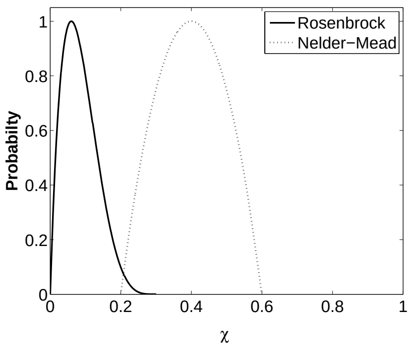


Fig. 4 Graphical Representation of the Probabilistic Scheme for Activating Local Searchers

registered in each direction. When such a condition is satisfied, the orthogonalization procedure of Gram and Schmidt is executed and the search, with the new set of directions, starts again. The algorithm is stopped when a budget condition is exceeded.

At the end of each DE generation, the following parameter is calculated:

$$\chi = \frac{\left| f_{\text{best}} - f_{\text{avg}} \right|}{\max \left| \text{f}_{\text{best}} - \text{f}_{\text{avg}} \right|_{\text{k}}}, \tag{8}$$

where f_{best} are f_{avg} are the fitness values of the best and average individuals of the population, respectively. $\max \left| f_{\text{best}} - f_{\text{avg}} \right|_k$ is the maximum difference observed (e.g. up to the *k*th generation), overall, beginning from the start of the optimization process. It is clear that χ varies between 0 and 1; it scores 1 when the difference between the best and average fitness is the largest observed, overall, and scores 0 when $f_{\text{best}} = f_{\text{avg}}$ i.e. all the population is characterized by a unique fitness value.

Besides considering it as a measurement of the fitness diversity (see [52] and [55]), χ is an estimation of the best individual performance with respect to the other

```
generate initial population pseudo-randomly;
apply PSO to the best performing individual;
replace the PSO result with the worst performing individual of the population;
while budget condition
  execute DE generation;
  compute $\chi = \frac{|f_{best} - f_{avg}|}{max|f_{best} - f_{avg}|_k}$;
  sample $\varepsilon \in [0,1]$
  if $\varepsilon < \frac{1}{B(2,2)} \cdot \frac{(\chi-a)^{(2-1)}(b-\chi)^{(5-1)}}{(b-a)^{(2+2-1)}}$
    execute NMA on an individual pseudo-randomly selected;
    replace the worst performing individual of the population with the NMA result;
  end-if
  if $\varepsilon < \frac{1}{B(2,5)} \cdot \frac{(\chi-a)^{(2-1)}(b-\chi)^{(5-1)}}{(b-a)^{(2+5-1)}}$
    execute RA on the individual having the best performance;
    replace the worst performing individual of the population with the RA result;
  end-if
end-while
```

Fig. 5 SFMDE pseudo-code

individuals. In other words, χ measures how much the super-fit outperforms the remaining part of the population. More specifically, the condition $\chi \approx 1$ means that one individual has a performance far above the average and thus one super-fit individual is leading the search. Conversely, the condition $\chi \approx 0$ means that the performance of the individuals are comparable and there is no super-fit. As a general guideline, a DE population that contains a super-fit individual needs to exploit the direction offered by the super-fit in order to eventually generate a new individual that outperforms the super-fit. Conversely, a DE population made up of individuals with comparable fitness values requires that one individual that clearly outperforms the others is generated in order to have a good search lead.

The parameter χ is employed to perform coordination of the local searchers. According to our algorithmic design, the SFMDE adaptation (and coordination of the local searchers) is based on an attempt to intelligently balance the DEs' need to generate a super-fit individual and prevent stagnation due to an excessive difference between the best performing individual and the others. More specifically, for each local searcher, a generalized beta distribution function is generated:

$$p(\chi) = \frac{1}{B(s,t)} \cdot \frac{(\chi - a)^{(s-1)} (b - \chi)^{(t-1)}}{(b-a)^{(s+t-1)}}, \tag{9}$$

where a and b are, respectively, the inferior and superior limits of the distribution; $B(s,t)$ is the beta function and s and t are the shape parameters. For the RA s and t, 2 and 5 have been set; for the NMA, 2 and 2. Both distribution functions have been normalized with respect to the maximum of the distribution in order to have a codomain in $[0,1]$. At each generation of the SFMDE, a value of χ is used for determining the probability of activating each of the local searchers. Each probability value is compared with a pseudo-random number generated between 0 and 1. If this number is lower than the probability value, the corresponding local search is performed. In addition, the RA is applied to the best performing individual of the population, while the NMA is applied to a pseudo-randomly selected individual. For both local searchers, the improved solution replaces the worst performing individual of the DE population. Figure 4 gives a graphical representation of the probability functions related to the local searcher activations. Figure 5 shows the SFMDE pseudo-code.

5 Numerical Results

In order to minimize the fitness in Eq. 4, the following algorithms have been applied. In SA has been run with an initial temperature equal to 1 and hyperbolical reduction of the temperature, as suggested in [57]. A standard Evolution Strategy (ES) on a population made up of 100 individuals has been run. A standard intermediate recombination has been chosen and the Gaussian mutation has been implemented resorting to the $1/5$ success rule. Finally, a $(\mu + \lambda)$ strategy has been chosen. A plain DE has been executed with the same parameter setting used for the

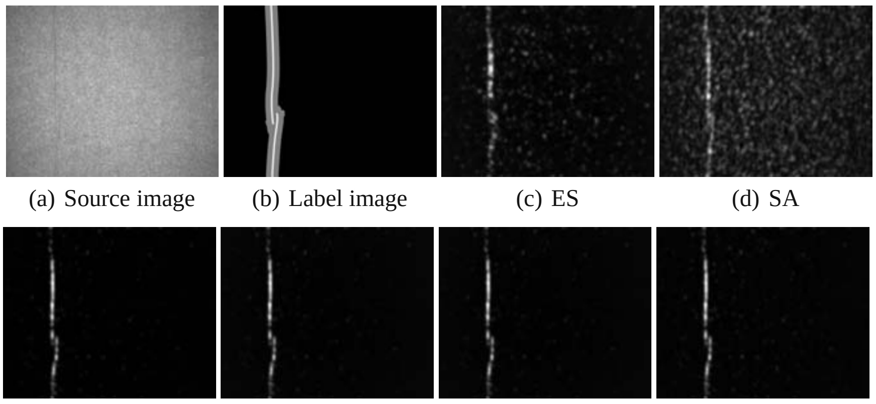
Table 2 Optimization results

	ES		SA		DE		MDE		EMDE		SFMDE	
	Fil 1	Fil 2	Fil 1	Fil 2	Fil 1	Fil 2	Fil 1	Fil 2	Fil 1	Fil 2	Fil 1	Fil 2
Weight	-5.893	-41.058	11.751	1.553	-1.507	94.995	21.479	91.568	21.459	91.568	82.701	-2.297
θ	4.330	0.408	0.643	1.613	3.530	1.519	2.135	4.658	2.135	4.658	3.104	2.919
ψ	4.552	5.099	0.819	3.318	4.912	3.122	6.042	3.153	6.042	3.153	3.104	4.725
σ_x	7.201	14.886	3.347	1.947	2.498	20	2.713	1.498	2.713	1.494	1.507	4.152
σ_y	1.008	10.905	6.023	7.128	1.512	20	3.609	20	3.611	20	20	20
λ	2.330	0.943	0.846	4.816	1.667	3.522	1.494	3.455	1.495	3.455	3.441	2.324
$f_{\min}$	-1.273		-1.361		-1.487		-1.509		-1.511		-1.521	
f_{mean}	-1.239		-0.978		-1.481		-1.493		-1.497		-1.500	
$f_{\max}$	-1.019		-0.732		-1.473		-1.339		-1.341		-1.466	
σ	0.0723		0.116		0.00382		0.0335		0.0336		0.0081	

DE framework. These benchmark algorithms have been compared with the MDE, EMDE and SFMDE. The MDE [45], EMDE and SFMDE have been run with a population size $S_{\text{pop}} = 100$. In the case of the SFMDE, the PSO has been run for 800 fitness evaluations on a population of 20 individuals with a maximum velocity equal to 15.5. The other local searchers have been run (as in the case of the EMDE) for 500 fitness evaluations; with reference to Eq. 9, $a = 0$ and $b = 0.6$ for the NMA and $a = 0$ and $b = 0.2$ for the RA.

Each algorithm has been run 30 times for $100,000$ fitness evaluations. The best solutions detected (parameters of the two filters, Fil 1 and Fil 2, respectively) by each algorithm are listed in Table 2. Moreover, the values of minimum, mean and maximum fitness and the related standard deviation values are also shown.

Figure 6 shows the functioning of the filters obtained by the optimization algorithms under analysis for an image belonging to the training set, while Fig. 7 shows the functioning for an image not belonging to the training set.


(a) Source image (b) Label image (c) ES (d) SA

(e) DE (f) MDE (g) EMDE (h) SFMDE

Fig. 6 Best results on a training set image

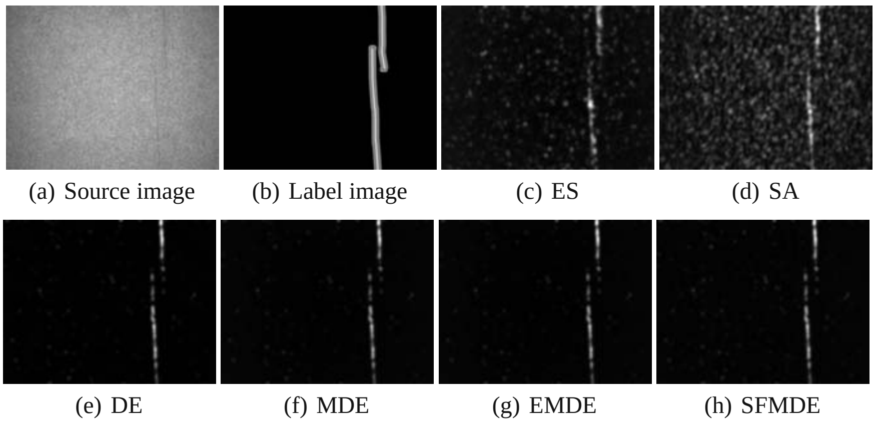

(a) Source image (b) Label image (c) ES (d) SA
(e) DE (f) MDE (g) EMDE (h) SFMDE

Fig. 7 Best results on a non-training test image

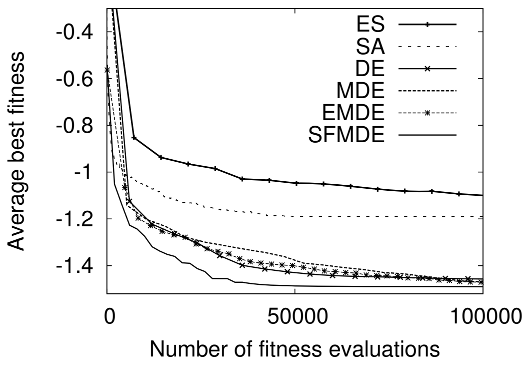


Fig. 8 Algorithmic performance

Figure 8 shows the average performance over the 30 experiments for the algorithms under scrutiny.

6 Conclusion

This chapter describes three different memetic algorithms employing a DE framework for filter design for defect detection in paper production.

Numerical results show that the algorithms based on a DE framework outperform other popular metaheuristics considerably for the image processing problem under study. Moreover, the MDE, despite inferior performance in terms of convergence velocity, outperformed the plain DE in terms of the final result. The EMDE slightly outperformed both the plain DE and the MDE in terms of the quality of the final solution and still seemed to behave reasonably well in terms of convergence velocity. The SFMDE seems to be clearly the most promising for this class of problems since it outperforms all other algorithms in terms of quality of the final solution it was able to find. In addition, Fig. 8 shows that the SFMDE has much better performance than all the other algorithms in terms of convergence velocity.

It can be seen that, unlike the MDE and EMDE, the SFMDE reached, during all 30 runs, reasonably good results after only $30,000$ fitness evaluations. The high-quality performance in terms of convergence velocity is a remarkable property of the SFMDE for this application, since it allows a better match with the industrial demand of acquiring a quite efficiently tailored filter bank in a reasonably short training time (about 20 minutes on a modern home computer and less than 5 minutes on a workstation).

References

1. Smith, R.D. (ed.): Roll and Web Defect Terminology. TAPPI Press (1995)
2. Jain, A.K.: Fundamentals of Digital Image Processing. Prentice Hall, Englewood Cliffs (1989)
3. Chang, C.Y.: Contextual-based Hopfield neural network for medical image edge detection. Optical Engineering 45(3), 37006 (2006)
4. Valli, G., Poli, R., Cagnoni, S., Coppini, G.: Neural networks and prior knowledge help the segmentation of medical images. Journal of Computing and Information Technology 6(2), 117–133 (1998)
5. Poli, R.: Genetic programming for feature detection and image segmentation. In: Fogarty, T.C. (ed.) AISB-WS 1996. LNCS, vol. 1143, pp. 110–125. Springer, Heidelberg (1996)
6. Cagnoni, S., Dobrzeniecki, A., Poli, R., Yanch, J.: Genetic-algorithm-based interactive segmentation of 3d medical images. Image and Vision Computing Journal 17(12), 881–896 (1999)
7. Hernandez, B., Olague, G., Hammoud, R., Trujillo, L., Romero, E.: Visual learning of texture descriptors for facial expression recognition in thermal imagery. Computer Vision and Image Understanding 106(2–3), 258–269 (2007)
8. Olague, G., Fernandez, F., Pérez, C., Lutton, E.: The infection algorithm: An artificial epidemic approach for dense stereo correspondence. Artificial Life 12(4), 593–615 (2006)
9. Trujillo, L., Olague, G.: Synthesis of interest point detectors through genetic programming. In: Keijzer, M., et al. (eds.) Proceedings of the 8th annual conference on Genetic and evolutionary computation, vol. 1, pp. 887–894. ACM Press, New York (2006)
10. Chan, C., Pang, G.K.H.: Fabric defect detection by Fourier analysis. IEEE Transactions on Industry Applications 36(5), 1267–1276 (2000)
11. Parker, S., Chan, J.: Dirt counting in pulp: An approach using image analysis methods. In: Younan, N. (ed.) Signal and Image Processing (2002) paper code 359–131
12. Iivarinen, J., Pakkanen, J., Rauhamaa, J.: A SOM-based system for web surface inspection. In: Machine Vision Applications in Industrial Inspection XII, SPIE, vol. 5303, pp. 178–187 (2004)
13. Dunn, D., Higgins, W.E.: Optimal Gabor filters for texture segmentation. IEEE Transactions on Image Processing 4(7), 947–964 (1995)
14. Sun, Z., Bebis, G., Miller, R.: On-road vehicle detection using evolutionary gabor filter optimization. IEEE Transactions on Intelligent Transportation Systems 6(2), 125–137 (2005)

15. Daugman, J.: Uncertainty relation for resolution in space, spatial frequency, and orientation optimized by two-dimensional visual cortical filters. Journal of the Optical Society of America A 2(7), 1160–1169 (1985)
16. Tsa, D.M., Wu, S.K.: Automated surface inspection using Gabor filters. The International Journal of Advanced Manufacturing Technology 16(7), 474–482 (2000)
17. Weldon, T.P., Higgins, W.E., Dunn, D.F.: Efficient Gabor filter design for texture segmentation. Pattern Recognition 29(12), 2005–2015 (1996)
18. Ji, Y., Chang, K.H., Hung, C.-C.: Efficient edge detection and object segmentation using Gabor filters. In: ACM-SE 42: Proceedings of the 42nd annual Southeast regional conference, pp. 454–459. ACM Press, New York (2004)
19. Ilonen, J., Kämäräinen, J., Kälviäinen, H.: Efficient computation of Gabor features. Research Report 100, Lappeenranta University of Technology, Department of Information Technology (2005)
20. Kumar, A., Pang, G.: Defect detection in textured materials using Gabor filters. IEEE Transactions on Industry Applications 38(2), 425–440 (2002)
21. Storn, R., Price, K.: Differential evolution - a simple and efficient adaptive scheme for global optimization over continuous spaces. Tech. Rep. TR-95-012, ICSI (1995)
22. Lampinen, J., Zelinka, I.: On stagnation of the differential evolution algorithm. In: Osmera, P. (ed.) Proceedings of 6th International Mendel Conference on Soft Computing, pp. 76–83 (2000)
23. Lopez Cruz, I.L., Van Willigenburg, L., Van Straten, G.: Parameter control strategy in differential evolution algorithm for optimal control. In: Hamza, M.H. (ed.) Proceedings of the IASTED International Conference Artificial Intelligence and Soft Computing, pp. 211–216. ACTA Press (2001)
24. Liu, J., Lampinen, J.: On setting the control parameter of the differential evolution algorithm. In: Proceedings of the 8th International Mendel Conference on Soft Computing, pp. 11–18 (2002)
25. Liu, J., Lampinen, J.: Adaptive parameter control of differential evolution. In: Proceedings of the 8th International Mendel Conference on Soft Computing, pp. 19–26 (2002)
26. Liu, J., Lampinen, J.: A fuzzy adaptive differential evolution algorithm. In: Proceedings of the 17th IEEE Region 10 International Conference on Computer, Communications, Control and Power Engineering, vol. I, pp. 606–611 (2002)
27. Liu, J., Lampinen, J.: A fuzzy adaptive differential evolution algorithm. Soft Computing - A Fusion of Foundations, Methodologies and Applications 9(6), 448–462 (2005)
28. Ali, M.M., Törn, A.: Population set based global optimization algorithms: Some modifications and numerical studies. Computers and Operations Research (31), 1703–1725 (2004)
29. Tvrdík, J.: Differential evolution: Competitive setting of control parameters. In: Proceedings of the International Multiconference on Computer Science and Information Technology, pp. 207–213 (2006)
30. Zaharie, D.: Critical values for control parameters of differential evolution algorithm. In: Matuŝek, R., Oŝmera P. (eds.) Proceedings of 8th International Mendel Conference on Soft Computing, pp. 62–67 (2002)
31. Zielinskiand, K., Weitkemper, P., Laur, R., Kammeyer, K.-D.: Parameter study for differential evolution using a power allocation problem including interference cancellation. In: Proceedings of the IEEE Congress on Evolutionary Computation, pp. 1857–1864 (2006)
32. Chiou, J.-P., Wang, F.-S.: A hybrid method of differential evolution with application to optimal control problems of a bioprocess system. In: The 1998 IEEE International Conference on Evolutionary Computation Proceedings, pp. 627–632 (1998)

33. Chiou, J.-P., Wang, F.-S.: Hybrid method of evolutionary algorithms for static and dynamic optimization problems with application to a fed-batch fermentation process. Computers and Chemical Engineering 23(9), 1277–1291 (1999)
34. Mydur, R.: Application of evolutionary algorithms and neural networks to electromagnetic inverse problems. Master's thesis, Texas A and M University, Texas, USA (2000)
35. Rogalsky, T., Derksen, R.W.: Hybridization of differential evolution for aerodynamic design. In: Proceedings of the 8th Annual Conference of the Computational Fluid Dynamics Society of Canada, pp. 729–736 (2000)
36. Wang, F.-S., Jang, H.-J.: Parameter estimation of a bioreaction model by hybrid differential evolution. In: Proceedings of the IEEE Congress on Evolutionary Computation, vol. 1, pp. 410–417 (2000)
37. Hendtlass, T.: A combined swarm differential evolution algorithm for optimization problems. In: Monostori, L., Váncza, J., Ali, M. (eds.) IEA/AIE 2001. LNCS, vol. 2070, pp. 11–18. Springer, Heidelberg (2001)
38. Lin, Y.-C., Wang, F.-S., Hwang, K.-S.: A hybrid method of evolutionary algorithms for mixed-integer nonlinear optimization problems. In: Proceedings of the IEEE Congress on Evolutionary Computation, vol. 3, pp. 2159–2166 (1999)
39. Lin, Y.-C., Hwang, K.-S., Wang, F.-S.: Co-evolutionary hybrid differential evolution for mixed-integer optimization problems. Engineering Optimization 33(6), 663–682 (2001)
40. Su, C.-T., Lee, C.-S.: Network reconfiguration of distribution systems using improved mixed-integer hybrid differential evolution. IEEE Transactions on Power Delivery 18(3), 1022–1027 (2003)
41. Lin, Y.-C., Hwang, K.-S., Wang, F.-S.: Hybrid differential evolution with multiplier updating method for nonlinear constrained optimization. In: Proceedings of the IEEE Congress on Evolutionary Computation, vol. 1, pp. 872–877 (2002)
42. Chiou, J.-P., Chang, C.-F., Su, C.-T.: Ant direction hybrid differential evolution for solving large capacitor placement problems. IEEE Transactions on Power Systems 19(4), 1794–1800 (2004)
43. Neumann, D., de Araujo, H.X.: Hybrid differential evolution method for the mixed H2/H robust control problem under pole assignment. In: Proceedings of the 44th IEEE Conference on Decision and Control and European Control Conference, pp. 1319–1324 (2005)
44. Kaelo, P., Ali, M.M.: Differential evolution algorithms using hybrid mutation. Computational Optimization and Applications 37(2), 231–246 (2007)
45. Tirronen, V., Neri, F., Kärkkäinen, T., Majava, K., Rossi, T.: A memetic differential evolution in filter design for defect detection in paper production. In: Giacobini, M. (ed.) EvoWorkshops 2007. LNCS, vol. 4448, pp. 320–329. Springer, Heidelberg (2007)
46. Hooke, R., Jeeves, T.A.: Direct search solution of numerical and statistical problems. Journal of the ACM 8, 212–229 (1961)
47. Hoos, H.H., Stützle, T.: Stochastic Local Search Foundations and Applications. Morgan Kaufmann, San Francisco (2004)
48. Cerny, V.: A thermodynamical aprroach to the traveling salesman problem. Journal of Optimization, theory and Application 45(1), 41–51 (1985)
49. Caponio, A., Neri, F., Tirronen, V.: Super-fit Control Adaptation in Memetic Differential Evolution Frameworks. Soft Computing-A Fusion of Foundations, Methodologies and Applications 13(8), 811–831 (2009)
50. Price, K.V., Storn, R., Lampinen, J.: Differential Evolution: A Practical Approach to Global Optimization. Springer, Heidelberg (2005)

51. Neri, F., Cascella, G.L., Salvatore, N., Kononova, A.V., Acciani, G.: Prudent-daring vs tolerant survivor selection schemes in control design of electric drives. In: Rothlauf, F., Branke, J., Cagnoni, S., Costa, E., Cotta, C., Drechsler, R., Lutton, E., Machado, P., Moore, J.H., Romero, J., Smith, G.D., Squillero, G., Takagi, H. (eds.) EvoWorkshops 2006. LNCS, vol. 3907, pp. 805–809. Springer, Heidelberg (2006)
52. Caponio, A., Cascella, G.L., Neri, F., Salvatore, N., Sumner, M.: A fast adaptive memetic algorithm for on-line and off-line control design of pmsm drives. IEEE Transactions on System Man and Cybernetics-part B 37(1), 28–41 (2007)
53. Neri, F., Mäkinen, R.A.E.: Hierarchical evolutionary algorithms and noise compensation via adaptation. In: Yang, S., Ong, Y.-S., Jin, Y. (eds.) Evolutionary Computation in Dynamic and Uncertain Environments. Studies in Computational Intelligence, pp. 345–369. Springer, Heidelberg (2007)
54. Neri, F., Toivanen, J., Mäkinen, R.A.E.: An adaptive evolutionary algorithm with intelligent mutation local searchers for designing multidrug therapies for HIV. Applied Intelligence 27(3), 219–235 (2007)
55. Neri, F., Toivanen, J., Cascella, G.L., Ong, Y.S.: An adaptive multimeme algorithm for designing HIV multidrug therapies. IEEE/ ACM Transactions on Computational Biology and Bioinformatics 4(2), 264–278 (2007)
56. Kirkpatrick, S., Gelatt, C.D.J., Vecchi, M.P.: Optimization by simulated annealing. Science (220), 671–680 (1983)
57. Szu, H., Hartley, R.: Fast simulated annealing. Physics Letters A 122, 157–162 (1987)
58. Balakrishnan, N., Basu, A.P.: The Exponential Distribution: Theory, Methods, and Applications. Gordon and Breach (1996)
59. Eberhart, R.C., Kennedy, J.: A new optimizer using particle swarm theory. In: Proceedings of the Sixth International Symposium on Micromachine and Human Science, pp. 39–43 (1995)
60. Kennedy, J., Eberhart, R.C.: Particle swarm optimization. In: Proceedings of IEEE International Conference on Neural Networks, pp. 1942–1948 (1995)
61. Nelder, A., Mead, R.: A simplex method for function optimization. Computation Journal 7, 308–313 (1965)
62. Lagarias, J.C., Reeds, J.A., Wright, M.H., Wright, P.E.: Convergence properties of the nelder-mead simplex method in low dimensions. SIAM Journal on Optimization 9, 112–147 (1998)
63. Rosenbrock, H.H.: An automatic method for finding the greatest or least value of a function. The Computer Journal 3(3), 175–184 (1960)

Fast Genetic Scan Matching in Mobile Robotics

Kristijan Lenac, Enzo Mumolo, and Massimiliano Nolich

Abstract. In this chapter, we address the problem of aligning two partially overlapping two-dimensional maps represented by data sets acquired using range sensors. The measured data may be incomplete and noisy. To solve this problem, we used a genetic algorithm for minimizing an alignment error. A lookup-based fitness function was devised. The considered range devices are laser and focalized ultrasonic scanners. Scan matching is often considered for mobile robot displacement and/or pose estimation tasks. We experimentally show that the algorithm is robust against noise and incomplete measurements and that it can be used for both the mentioned tasks. Moreover, the proposed algorithm is suitable for both local and global robot pose estimation. Experimental results related to the convergence, accuracy and speed of the proposed algorithm with different coding approaches are reported. We compare our approach with other scan matching algorithms proposed in the literature, and we show that our approach is faster and more accurate than the others.

1 Introduction

The matching and analysis of geometric shapes is an important problem that arises in various applications areas, particularly in computer vision, pattern recognition and robotics. In a typical scenario, we have two shapes and we want to find the optimal transformation (translation and rotation) that matches one shape with the other as accurately as possible. Shapes are typically represented by finite sets of points in two or three dimensions. An important area of matching of geometric shapes is

Kristijan Lenac
A.I.B.S. Lab S.r.l. - Via del Follatoio 12, Trieste, Italy
e-mail: `klenac@units.it`

Enzo Mumolo and Massimiliano Nolich
DEEI, University of Trieste - Via Valerio 10, Trieste, Italy
e-mail: `mumolo,mnolich@units.it`

scan matching applied to robot self-localization. Mobile robots, in fact, must have the ability to ascertain their pose (position and orientation) while navigating or exploring an environment.

Many localization techniques, such as odometry, GPS, inertial systems, range sensors, etc., are typically used in mobile robotics. A robot's position on the map is most easily tracked using dead-reckoning that determines the robot's location by integrating data from wheel encoders (that count the number of wheel rotations). In many cases, however, dead-reckoning fails to accurately position the robot because of many reasons, including wheel slippage. If the robot slips, the wheel rotation does not correspond to the robot's motion, and thus encoder data, which measures the state of the wheel rotation, does not reflect the robot's net motion, thereby causing positioning error. Global Positioning Systems (GPS) offer an alternative to dead-reckoning, but GPS signals may not be available, e.g. in indoor environments. High-performance inertial systems, on the other hand, are very expensive. Landmark-based localization can be a better choice, but the environments may be unknown and not structured. Therefore, a precise and stable localization becomes a challenging problem when the robot experiences unacceptable positioning error with the odometry, does not have an external positioning device such as GPS and moves in an unknown environment with no artificial or natural landmarks.

In this chapter, we deal with localization methods based on matching of scan data obtained from a range device, namely, a laser range finder (LRF) or rotating ultrasonic range devices, using genetic algorithms. Each scan provides a contour of the surrounding environment. A new genetic scan matching algorithm is presented, called GLASM, with the following general characteristics.

- The fitness function we developed does not need to compute any correspondence.
- The best results are obtained with Gray coding of the chromosome and with binary representation.
- In general, the success ratio is better than classical methods such as Iterative Closest Point algorithm (ICP) [1] and genetic-based matching techniques [2].
- The algorithm is very fast mainly because it is based on a lookup table; moreover, the fitness computational complexity is $O(N)$, where N is the number of scan points.

Generally speaking, there are two different approaches to determine the robot's pose with scan matching: relative and absolute. In relative approaches—or movement estimation approaches—given an arbitrary initial pose, e.g. $p_0 = (0,0,0)$, the robot's current pose relative to p_0 is incrementally updated when the robot is moving. Let S be the current scan and R the reference scan, e.g. a scan acquired before S. If both scans have been acquired from different robot poses, the transformation that maps S onto R is calculated, which corresponds to the movement of the robot between the scans. In absolute approaches—or position estimation approaches—on the other hand, the position and orientation of the robot within an a priori given map or a known map is calculated.

Based upon the availability of an approximate alignment of two scans prior to performing the matching, scan matching can be further classified into local and global. Local scan matching is when an Initial Position Estimate (IPE) is available from which the search is initiated. It is generally used for robot position tracking, where some other localization methods such as odometry provide an IPE. The provided IPE has typically a small localization error, and local scan matching is used to further reduce this error. On the other hand, with global scan matching, we are able to generate and evaluate a position hypothesis independently from initial estimates, thus providing the capacity to correct position errors of arbitrary scale.

The structure of this chapter is as follows. Section 2 addresses the problem of scan matching in general, providing general definitions used in the rest of the chapter and briefly describes the sensors used in this work, in particular, ultrasonic-focused sonars and laser range sensors. Section 3 reports a brief review of related literature, while Sect. 4 describes the GLASM algorithm. In Sect. 5, it is compared with other classical approaches. Final remarks and conclusion are discussed in Sect. 6.

2 Scan Matching: Problem Formulation

The main issue in robot self-localization is how to match sensed data, acquired with devices such as laser range finders or ultrasonic range sensors, against reference map information. The reference map can be obtained from a previous scan or from an a priori known map. Usually, the problem is solved by defining a distance measure between the two scans and searching for an appropriate rigid transformation that minimizes the distance. In this chapter, we consider a two-dimensional (2D) case, i.e. we consider a mobile robot on a flat ground. Its pose is described by a triple (x,y,φ), where (x,y) and φ denote the robot position and orientation, respectively.

Scan matching can be described in an intuitive way by considering one point of the reference scan and one of the new scan under the assumption that they sample the same point in the environment but from different positions. Consider a point $P(x,y)$ in the (x,y) coordinate system and a point $P'(x',y')$ in the (x',y') coordinate system, as shown in Fig. 1. The two coordinate systems differ by a rotation φ and a translation, represented by a bias vector $\overline{b} = (b_x, b_y)$.

The problem consists of finding the two factors $\overline{b}$ and φ that make the two points P and P' correspond. Using a rotation/translation operator, the point P_i' is mapped onto the point P_i as follows:

$$\begin{cases} x_i = x_i'\cos(\varphi) - y_i'\sin(\varphi) + b_{i,x} \\ y_i = x_i'\sin(\varphi) + y_i'\cos(\varphi) + b_{i,y} \end{cases}$$

Normally, the φ and $\overline{b}$ terms are unknown and must be estimated. The estimation can be performed by minimizing the error between the ith points P_i and P_i':

$$\overline{E_i} = \begin{pmatrix} E_{i,x} \\ E_{i,y} \end{pmatrix} = \begin{pmatrix} x_i - x_i'\cos(\varphi) + y_i'\sin(\varphi) - b_{i,x} \\ y_i - x'\sin(\varphi) - y_i'\cos(\varphi) - b_{i,y} \end{pmatrix}$$

If only one point is considered, however, it is possible to find the optimum bias, but it is impossible to determine the rotation. For this purpose, the problem must

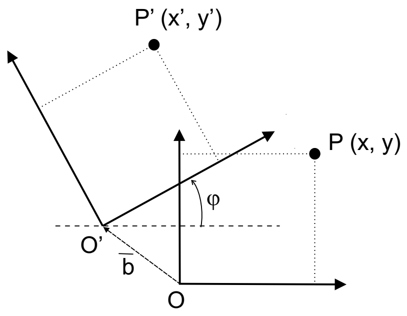


Fig. 1 Corresponding points of two different scans in different reference frames

be turned into a least square problem by considering a sufficient number of corresponding points, *N*. In this case, the square error is as follows:

$$E = \sum_{i=0}^{N-1} |E_i|^2 = \sum_{i=0}^{N-1} (E_{i,x}^2 + E_{i,y}^2)$$

Lu and Milios [3] have shown that the result of this minimization error is the following:

$$\phi = \arctan\left(\frac{S_{xy'} - S_{yy'}}{S_{xx'} + S_{yy'}}\right)$$

$$b_x = \overline{x'} - (\overline{x}\cos(\varphi) - y\sin(\varphi)), b_y = \overline{y'} - (\overline{x}\sin(\varphi) + \overline{y'}\cos(\varphi)),$$

where $\overline{x}$, $\overline{x'}$, $\overline{y}$ and $\overline{y'}$ are the point averages, and the *S* terms are covariances. This approach requires that exact correspondences of points are established. In practice, however, exact point correspondence is impossible to obtain due to many sources of noise, such as mechanical deformation during robot movement, terrain uneveness, occluded areas, sensor noise etc. Thus, a high degree of robustness of the scan matching algorithms is required to operate in real-life environments.

2.1 Definitions

Some terms used in this chapter are defined in this section.

- *Successful Matching:* a matching that results in an estimated position within the ellipsoid centered in the true position $(x_{\text{true}}, y_{\text{true}}, \theta_{\text{true}})$. A successful matching yields a position estimate close to the true position.
- *Success Ratio (SR):* ratio between the number of successful matchings (N_{Msucc}) and total matchings (N_{Mtot}) performed in one set of localization trials:

$$SR = \frac{N_{\text{Msucc}}}{N_{\text{Mtot}}}.$$

  It measures the ability of the algorithm to converge for a given environment and a set of scans.

- *Maximum Success Ratio ($SR_{\max}$):* a maximum possible success ratio for a given environment and a set of scan pairs. As a matter of fact, in the experiments, SR is rarely equal to 1. In other words, there is always a maximum success ratio $SR_{\max}$, $SR_{\max} \leq 1$, for a given environment and a set of scans regardless of the matching algorithm used.
- *Accuracy:* it is the absolute value of the difference of the estimated and the true position or rotation in the case of successful matching. It is easily determined for particular test conditions if the true scan positions are known. We use the average and the variance of the position estimate error for both position and rotation. In the experimental results reported further in Sect. 5, only successful matchings were used for the computation of accuracy. It is worth noting that an algorithm with a higher SR includes the computation of accuracy cases for which the others failed to converge.
- *Fitness function hit area:* in GLASM algorithm, the area in the lookup table including cells which are less distant than the distance threshold from a reference scan point or known map. These cells are marked for a fast lookup by the fitness function.

2.2 The Sensors Used in This Work

The scan devices used in this work are rotating-focused sonar devices, described in [4], and laser range sensors. Both systems are based on the time-of-flight range measurement principle. However, ultrasonic devices have some advantages over laser sensors: for example the signal is not hazardous for humans; they can be used in presence of smoke or fog and can detect transparent objects such as glasses. In any case, the sensor data must be processed to enhance the signal and extract the information content of the registration. On the other hand, lasers scan devices have better spatial resolution than ultrasonic devices due to the narrow beam. It is worth noting that good ultrasonic sensors have a beamwidth in the order of 10 degrees, and even the focalized beam described in [4] is about 4 degrees, which is much higher than a laser. As a result, ultrasonic readings are less dense than those of a laser. In general, the reflection characteristics are also better for lasers than sonars.

As an example, in the left panel of Fig. 2, we report an ultrasonic scan realized with the rotating device of [4] matched against an a priori map, while in the right panel, we report similar results for a laser device of the Sick laser measurement systems family.

Once the correct functioning of the algorithm with the two types of sensors is established, the results reported hereafter are based on real acquisitions from the rotating-focused sonar device described in [4].

3 Related Work

The literature on scan matching is very large. In this section, we report a brief description of some known approaches which, although highly incomplete, put the

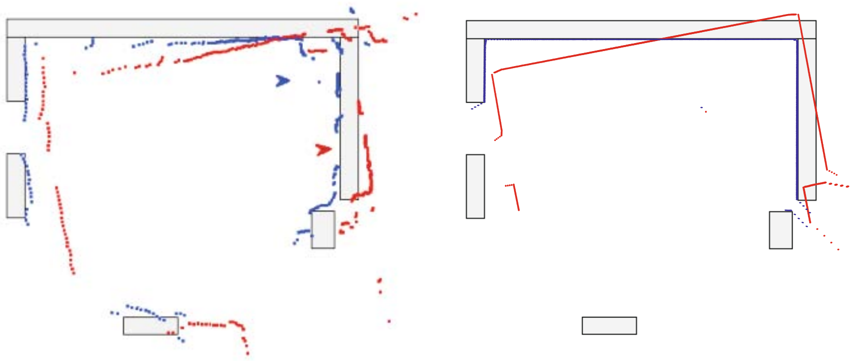


Fig. 2 Scan matching performed on scans obtained with rotating ultrasonic focused sensor (*left*) and laser scanner (*right*). The points in light grey represent raw data and those in dark grey represent the aligned data

results presented in this chapter in the correct context. Although other taxonomies may be used, here, we classify scan matching approaches in those based on correspondences and those not requiring correspondences.

3.1 Algorithms Based on Correspondences

Scan matching algorithms based on correspondences can be categorized as

- *Feature-to-feature correspondences*. In this case, features such as line segments, corners or range extrema [5] are extracted from the actual and reference scans and matched between them. Of course, these approaches require the environment to be structured or to contain the considered features.
- *Point-to-feature correspondences*. The points of the scan are matched to features such as lines, which can be part of an a priori known map [6]. Other authors extract features obtained with certain signal processing. Biber and Strasser [7] considered Gaussian distributions with mean and variances computed from the point falling into cells of the grid. Also, these approaches require the presence of the chosen features in the environment.
- *Point-to-point correspondences*. Widely used heuristic methods for aligning 2D or 3D point sets are variations of the ICP proposed by Besl and McKay in [1] and then in [8, 9], to cite a few. Basically, ICP has three basic steps: first, pair each point of the first set to the second one using a corresponding criterion; second, compute the rotation and translation transformations that minimize the mean square error between the paired points and finally, apply the transformation to the first set. The optimum matching is obtained by iterating the three basic steps. However, ICP has several drawbacks. First, its proper convergence is not guaranteed, as the problem is characterized by local minima. Second, ICP requires a good pre-alignment of the views to converge to the best global solution.

Matching points-to-points is the most general approach that does not require features to be present in the environment. Lu and Milios [3] were the first to apply this approach for localization of mobile robots. They use ICP to compute translations and propose techniques for the selection of corresponding points. Pfister et al. [10] developed the WRSM method, which extends the approach of [3] by also considering the uncertainty of the estimated motion in order to integrate the scan matching-based motion with odometry information.

3.2 Algorithms Which Do Not Require Correspondences

In [11], the solution is searched by performing gradient descent on a score function. Such a function is built by convolving the reference scan with a Gaussian kernel and then correlating it with the sensor scan. This approach has been subsequently refined in [7] by defining a closed form for a potential function that can be minimized using the Newton algorithm. In [12], a correlation-based approach that is well suited in polygonal environments is presented. The orientation of each scan point is found and the circular histograms of these are build. Then, the rotation is found by correlating the normal histograms. In [13], global localization is performed by using a 2D correlation operator. This method evaluates every point in the search space and is therefore very robust to noise. It is a global, multi-modal, non-iterative matcher that can work in unstructured environments.

3.3 Use of Stochastic Optimization

The initial optimization-based techniques proposed for point matching methods were based on ICP, which is a direct-descent technique, followed by gradient computation approaches [14]. Such approaches are based on calculating gradients of either probabilistic or squared-error [15, 16] cost functions. Amongst them, the ICP procedure was the most popular because it is simple and effective; thus, many variants have been proposed as described in [17]. Recently, stochastic optimization approaches such as evolutionary programming [18], simulated annealing [19] and genetic algorithms [2, 20–23] have been used. These methods introduce a stochastic component in the search for either the correspondence of points [18] or the whole pose [19–21].

4 GLASM

In this section we describe the proposed algorithm, which we called GLASM (Genetic Lookup-based Algorithm for Scan Matching). It aims at finding the (x, y) translation and the rotation φ that produce the best alignment between two different scans. The algorithm is based on the computation of a lookup table that divides the plane of scan readings in a grid for a rough but fast reference point lookup as will be shown shortly in the description of the fitness function.

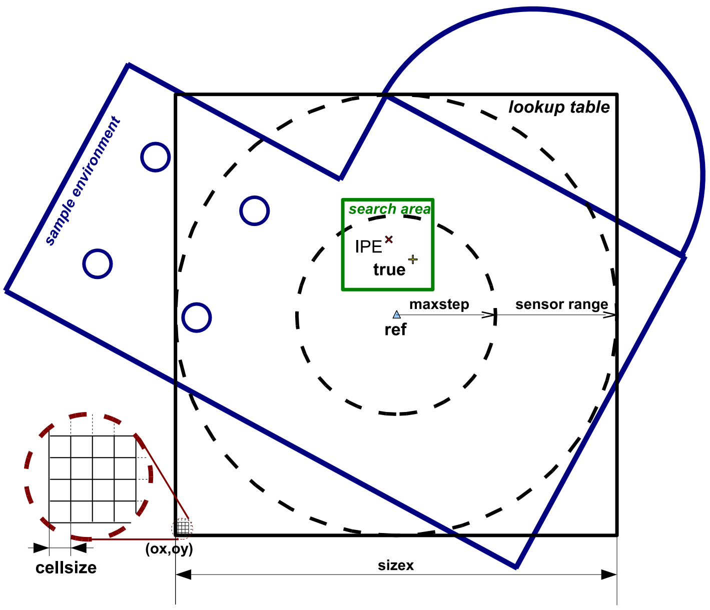


Fig. 3 Lookup table and search space size in local scan matching step

Each parameter of the scan position (x, y, φ) is coded in a chromosome as a string of bits as follows: *nbitx* for coding x, *nbity* for y and *nbitrot* for φ. The problem space is discretized in a finite solution space with a resolution that depends on the extension of the search area and the number of bits in each gene. The search area limits can be set based on the problem at hand. In the case of pose tracking where odometry measurements are available they are usually set on the basis of odometry error model.

The positional information is coded in the genes using the Gray code. The inverse Gray code value of the bit string is the scan position. In this way the variations in the bit string caused by the mutation or crossover operators produce proportional variations in the position they represent. Using a simple binary code a change in one bit may cause a significant variation in the position. We found that for the scan matching application, a simple binary code leads to a reduced efficiency of the genetic algorithm.

The genetic algorithm starts with a population of *Popsize* chromosomes randomly selected with uniform distribution in the search space. Each individual represents a single position of the new scan. An example of the matching process is shown in Fig. 4. The goal of the scan matching is to estimate the position of the new scan relative to a reference scan or a given map, which is 'best fitted' according to a fitness value.

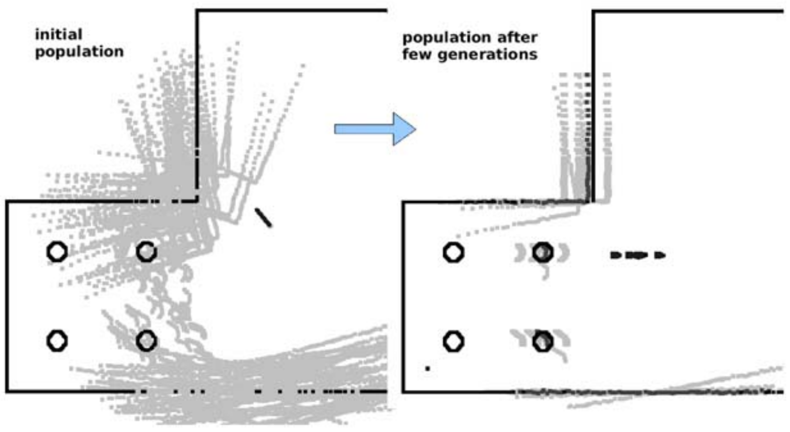


Fig. 4 The reconstruction of the scene

4.1 Fitness Function

Fitness computation must be done very quickly and it must provide a value that is closely correlated with the degree of matching between two scans. The GLASM fitness function is formulated by accumulating matching errors and normalizing them by the number of valid corresponding points. The most fitted values, those that point to a better overlap of two scans, are the positions with a smallest cumulative matching error.

As soon as the reference scan is available, a lookup table is created as depicted in Fig. 5. Each cell represents a portion of the 2D plane of scan readings. The overall dimensions of the lookup table cover all the search space. In the case of pose tracking, the lookup table is centred in the reference scan position. In order to cover all the possible sensor readings of the new scan, it has to be at least as large as the step size plus the sensor range. For position estimation with an a priori given map the lookup table should at least cover the map. Each cell in the table is marked with a boolean value 0 or 1. The cells of the table are initialized with 0 and only the cells close to a reference scan point are marked with 1.

The genetic algorithm evaluates the fitness value for a chromosome as follows: for each point of the new scan, a direct lookup is made in the lookup table to establish if the cell corresponding to the point is set to 1. The number of points of a new scan having a corresponding point in a reference scan is then taken as a fitness value for this chromosome, i.e. for the new scan evaluated in the position coded in a chromosome. This in fact is directly proportional to the overlapping of two scans, i.e. the degree of matching between the two.

This way, there is no need for the correspondence function since no pairings need to be done between scan points. There is also no need for the matching error function since the fitness value is directly obtained by counting positive lookup returns. The speed-up gain of our approach comes at the cost of building the initial lookup table

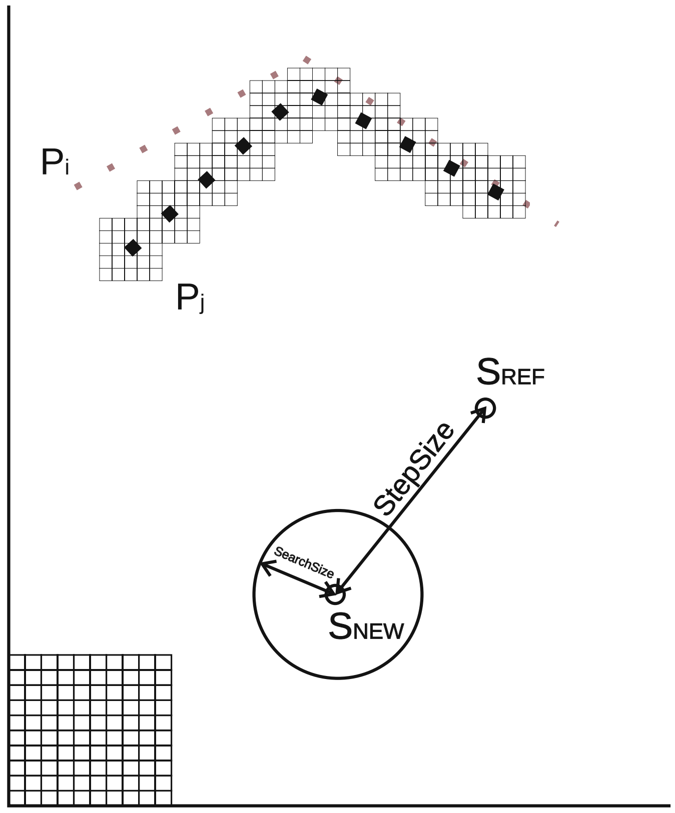


Fig. 5 The environment is discretized and a lookup table is built. The cells near the points of the reference scan are depicted. These cells are the only ones to be marked in the lookup table

and of a potential reduction of matching process' accuracy posed by quantization errors introduced by the lookup table. However, the experiments presented in this chapter show that both of them are negligible. In fact, the creation of the initial lookup table is very fast and the accuracy reduction is lower or comparable to the resolution of the finite solution space; therefore, the overall effect is negligible.

In order to avoid some areas carrying more points than others, the scans are resampled. This is because a scan of an object close to the sensor has denser readings than the one far away. The fitness function would otherwise count a higher score when overlapping dense areas containing more points. This would reduce the efficiency of the matching process.

On the basis of the above considerations, it is clear that we define our basic fitness function as the sum of the squares around each point coming from one scan that intercepts the point in the new scan:

$$f_{S_{\text{NEW}},S_{\text{REF}}} = \sum_{i=1}^{N} \rho(i), \tag{1}$$

where S_{NEW} and S_{REF} are the two scans to be matched, N is the number of points in S_{NEW} and the value of ρ is as follows:

$$\rho(i) = \begin{cases} 1 & \text{if a point of } S_{\text{NEW}} \text{ lies in the square around a point in } S_{\text{REF}} \\ 0 & \text{otherwise} \end{cases} \tag{2}$$

5 Experimental Results

We tested the GLASM algorithm with real scan data obtained with a focused ultrasonic sensor [4]. Furthermore, for an in depth analysis of success ratio, search space size, accuracy and speed of the algorithm, a scan matching simulator, expressly developed for that type of sensor, has been used. The use of a simulator is the only possible approach that allows to run a huge number of scan matching tests in a controlled testing environment with scans taken from arbitrary positions in arbitrary environments. The simulator also guarantees the knowledge of the true position from where the scan was made, thus enabling the exact estimation of the accuracy and the success ratio of the matching process. With real scans, the true position must be measured by hand or by a more accurate localization method and this is not a simple task and is error prone. With the use of a simulator, it is also possible to save the scans and repeat the same series of tests while varying the parameters of the algorithm and observing the effect on performance. This is especially important when performing comparisons with other scan matching methods. The algorithm's performance regarding the success ratio was investigated first. Recall that a matching is considered successful if the estimated scan position lies within a range from the true position. The radii of the ellipsoid in the implementation were 0.1 m on the x and y axis and 0.1 rad for the rotation, i.e. the matching is to be considered successful if the resulting position is at least 10 cm near the true position and rotated less than 0.1 rad from it. In order to obtain a success ratio value that is less dependent on a particular set of scan pairs used in the tests, different matchings from many randomly chosen scan pairs were performed. In this way, the success ratio may be considered a characteristic indicator of the algorithm's matching performance for that particular environment and sensor. For each obtained pair $(S^i_{\text{REF}}, S^i_{\text{NEW}})$, the new scan was further randomly displaced several times from the true position. That is because the IPE is the starting point for the iterative algorithms search, and since different IPEs for the same scan pair may converge or diverge differently, several randomly chosen IPEs were considered. The IPE is not used by genetic algorithms; however, the centre of the search space for genetic algorithms was placed in the IPE just like in real robot exploration. This implies that for different IPEs, the genetic algorithm searches in different areas. A single test cycle in the experiments therefore consisted of a total of $N_{\text{Mtot}} = N_{\text{pa}} \cdot N_{\text{ip}}$ matchings, with N_{pa} being the number of different randomly chosen scan matching pairs and N_{ip} different IPEs of the new scan. For each test cycle, the success ratio, accuracy and average matching time were calculated. The accuracy of the algorithm was calculated by measuring the average and variance of the displacements from the true position. The execution time was measured on a Pentium 4 HT, 2.8 GHz computer.

5.1 Determination of the Parameters of the Genetic Algorithm

The parameters of the genetic algorithm were chosen with the goal of maximizing the success ratio. A new test cycle was performed for different configurations of algorithm parameters. The mutation and crossover probability were then chosen from observation. The population size and the number of generations performed should be chosen as low as possible. The lower the population size and number of generations, the less computations must be performed and therefore the faster the algorithm. However, the success ratio for a given environment and search space size starts to drop when population size and generations become too low.

The algorithm's parameters are summarized in Table 1.

Table 1 The values of GLASM parameters used in the experiments

Genetic parameters		Lookup table	
Max generations	Variable with search space size	Lookup table size	Variable with search space size
Population size	Variable with search space size	Cell size	2×2 cm
Chromosome length (X, Y, θ)	6 + 6 + 6 = 18 bit	Fitness function hit area (distance from ref. scan point or map)	9 cm
Crossover probability	1		
Mutation probability	0.00925926		

5.2 Implementation Details

Different GLASM implementations have been tested and compared with each other. A comparison between Gray and Simple Binary encoding variants was performed. In the same test conditions and for the success ratio of $SR \geq 90\%$, the Gray variant required less generations and a smaller population for the same performance. Moreover, the repeatability of the performance is better. Furthermore, there is no perceivable gain in the speed for the Simple Binary variant since the conversion in the Gray code is memory mapped with negligible computation time Fig. 6. Therefore, a Gray variant has been chosen for the experiments presented in the following.

5.3 Comparison

The GLASM algorithm has been compared to a pure 2D point-to-point version of the ICP algorithm (without tangent line calculations), as reported in [21]. Two versions of the most computationally expensive part, the correspondence search algorithm, were used and compared. The first one is a classical implementation which

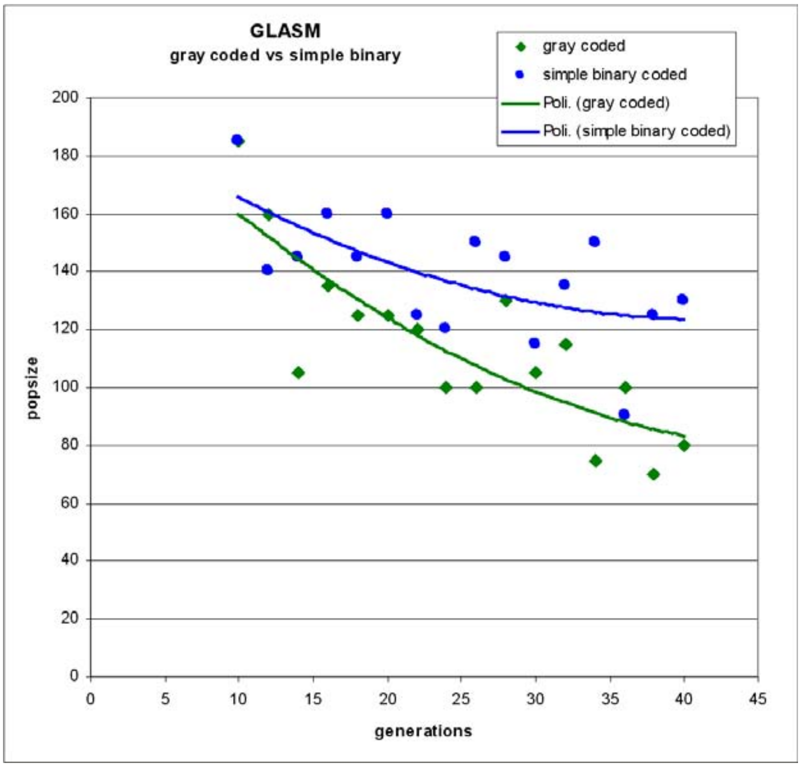


Fig. 6 Comparison between a simple binary and Gray coded GLASM. Population size and number of generations that yielded success ratio above 90% for the same environment and set of scan pairs

computes the squared distances for every possible combination of reference and current scan points; the second is an optimized version [24] which yields a speedup of approximately 4*x* without any noticeable reduction of the algorithm's accuracy. The speed of this faster ICP implementation is reported later in the document and compared with GLASM. Special care was taken to implement the best and highly optimized versions of the algorithms.

5.4 Search Space Size Considerations

For a given set of scan pairs and search space size, iterative correspondence point algorithms [1, 10] do not have the flexibility of varying some algorithm parameters in order to perform a more thorough search when necessary. Genetic algorithms, on the other hand can accomplish better success ratio and accuracy by increasing the population size and generations. The success ratio eventually converges towards the maximum success ratio for that environment and that particular set of scans.

Table 2 Comparison of GLASM with ICP for local scan matching

			Position error		Rotation error	
	SR	time (ms)	avg (m)	var	avg (rad)	var
ICP	87%	103	0.01500	0.00430	0.00019	0.00002
GLASM	92%	22	0.04000	0.01100	0.00042	0.00012
ICP	79%	159	0.01900	0.00715	0.00044	0.00008
GLASM	93%	23	0.04300	0.01068	0.00045	0.00010

Genetic algorithms start the search process by distributing an initial population in a given search space. Iterative algorithms based on point correspondences on the other hand, start the search process from the IPE, which must be available. Depending on how far and how misaligned this estimate is, they converge or fail to do so. Therefore, the performance of these algorithms is satisfactory only inside an area close to the true position (approx < 1.0 m distance and ± 0.4 rad rotation).

ICP is therefore a local matcher and is quite different from GLASM, which can be used for both global and local scan matching and can scale easily with different search space sizes. Nevertheless, since ICP is wildely, used in local scan matching and its performance is well known, it is interesting to compare the accuracy, speed and success ratio of the two algorithms for local scan matching with small search space size where ICP performs well.

Table 2 shows the results of a test cycle of 450 matchings performed for two different search space sizes inside the convergence region of the ICP algorithm. In the first case, random positions were placed on the circle distant 0.2 m from true position, and the rotation was chosen $+0.2$ rad or -0.2 rad at random. The test was then repeated for 0.4 m and 0.3 rad range. In both cases, GLASM has a better success ratio than ICP. This is true in general and has been verified for other environments and search space sizes. ICP has a slightly better accuracy than GLASM, which was expected. In a small search space, a reduced population size and number of generations are used. In fact, for a small search space size, GLASM achieves high success ratios with a population size of 30 and after only five generations. Moreover, GLASM is limited by the discretization of the GLASM lookup table and the size of the fitness function hit area. As for the speed, for a fixed population size and a number of generations, the genetic algorithms have constant processing time regardless of the convergence. For iterative algorithms, the speed varies between iterations due to correspondence points pairings and number of performed iterations. The stop criteria is set when the convergence slows down to a point that there is little variation between successive iterations. More precisely, the process stops if at least five initial iterations have been performed, and in the last three iterations, rotational and position displacements are less than a threshold; otherwise, it continues until a maximum of 50 iterations. It is clear that ICP's failure to converge costs time. The results show that GLASM is at least five times faster than ICP while still having a better success ratio. This is quite remarkable since ICP is considered a very fast algorithm.

5.5 Comparison with a Classical Genetic Scan Matching Technique

GLASM has also been compared to an existing 2D genetic optimization technique for scan matching, the genetic algorithm based on GCP transform proposed by Yamany et al. [2], called Yamany Genetic Matching (YGM), which is classical in the area. The YGM's fitness function is based on corresponding point pairings between a reference scan and new scan point. To speed up the fitness evaluation, the Correspondence Grid is initially created with a discretization of the plane containing the scan points, which calculates the displacement vectors from each cell to the nearest reference scan point. That way, instead of performing a search for correspondences for each fitness evaluation, a simple lookup in the Correspondence Grid yields the displacement vector. The initial cost of building the Correspondence Grid is small compared to a high number of fitness evaluations performed during a matching process, which is usually the case.

The main differences of the two genetic algorithms are as follows.

1. The YGM transform calculates a displacement vector, while GLASM does not.
2. This calculation is done for each cell of the grid, while GLASM only marks (without calculation) a very limited number of cells around reference scan points (model set).
3. The YGM transform requires a higher grid resolution for a good approximation of the closest point. On the contrary, GLASM does not calculate a displacement vector, and so it does not suffer from displacement vector quantization error.
4. GLASM does not use the correspondence function and matching error function, and so the fitness value is provided by direct count (direct lookup). A simple fitness function assures faster and straightforward calculation.

This time a medium and large search space sizes were used in the experiments (Fig. 7, Table 3); therefore, a larger number of generations and a bigger population size were necessary to maintain a high success ratio for the environment.

The test results have shown that under the same test conditions (same environment, set of scan pair positions, scans) and GLASM yields a higher success ratio for the same number of generations and population size, i.e. it converges in cases where YGM does not. The accuracy of the algorithms is analyzed in Fig. 8. Each darker point in the graphs corresponds to the average error in position (left graph) and rotation (right graph) for a series of 150 matchings (one test cycle), where the lighter points are the variance. Only successful matchings are considered in computing the averages and variances.

The average position and rotation errors are more or less constant with population size, but decrease with the number of evolved generations. For GLASM, the average position error finally settles to a distance of approximately 3.4 cm from the true position and rotation error to about 0.00027 rad from the true scan angle. YGM settles down to 2.9 cm and 0.00048 rad.

GLASM and YGM clearly differ as to the variance. As shown in Fig. 8, GLASM has a lower variance than YGM, and these experiments present a great repeatability.

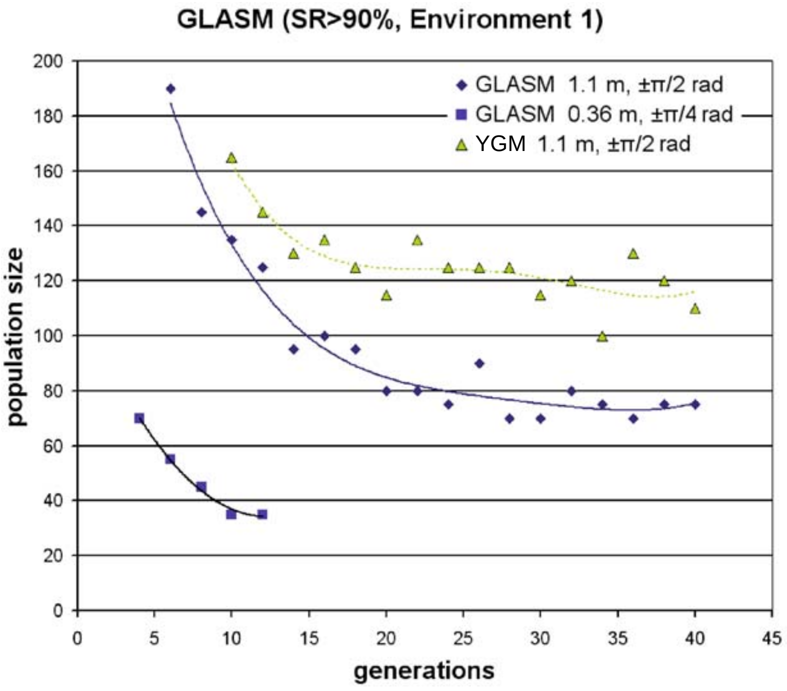


Fig. 7 The number of generations and a population size needed to reach a success ratio above 90%. Two series of data are plotted for GLASM algorithm corresponding to a search space size of ($\pm$0.36 m,$\pm\pi$/4 rad) and ($\pm$1.1 m,$\pm\pi$/2 rad). One data series (in *lighter colour*) is plotted for YGM algorithm for the ($\pm$1.1 m, $\pm\pi$/2 rad) case

These characteristics lead to a feasible implementation of GLASM even with strict computational time constraints.

Let us now analyze the speed of the algorithms. Both algorithms speed up the fitness function evaluation by building a lookup-based grid in memory as soon as the reference scan is known. Table 3 compares the time spent to prepare the grid. Two sizes of the grid are considered: one suitable for a medium size environment (up to 100 m^2), and the other for a larger environment (up to 250 m^2).

The overall processing time is thus a sum of the time spent in the preparation of the correspondence grid, which is performed only once for a single reference scan or known reference map and the processing time of the genetic algorithm for the matching of the new scan.

In [2], it is suggested that the speed may be improved by selecting small cell size in the region directly surrounding the model set and a slightly larger value for the rest of the grid. In typical robotic applications and with reference to Fig. 3, the size of the grid is just slightly larger than the enclosed environment, and so the suggested approach could result in modest improvements in speed but with greater implementation complexity.

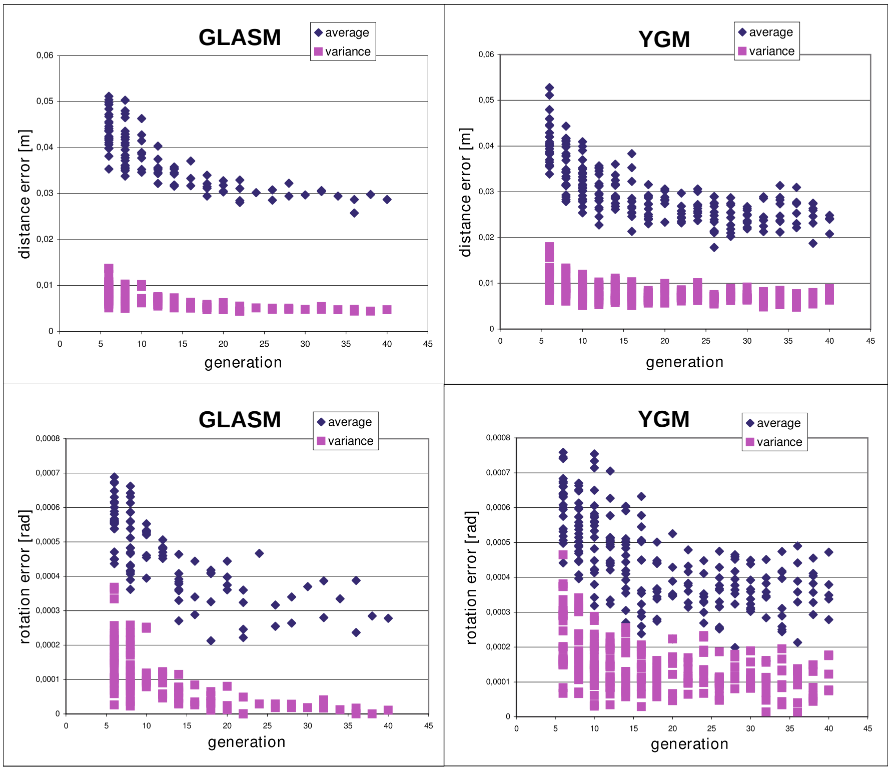


Fig. 8 Accuracy comparison between YGM and GLASM

When building a lookup table, GLASM only updates a very limited number of cells around reference scan points (fitness function hit area), leaving the vast majority of cells intact. The fitness function evaluation is also much faster since GLASM does not search for correspondences and it does not compute a matching error with the set of found corresponding points.

The average execution time for a single matching (test cycles with $SR \geq 90\%$) is: GLASM: 350 ms and YGM: 461 ms. This is an improvement of 24% considering only the matching phase.

Table 3 Average time spent in the preparation of the correspondence grid (YGM) and lookup table (GLASM)

	Preparation of the correspondence grid	
	(medium size environment < 100 m^2) 1000×1000 cells	(large environment < 250 m^2) 5000×5000 cells
YGM	19,000 ms	several minutes
GLASM	4 ms	110 ms

6 Final Remarks and Conclusions

In this chapter, we have described and discussed a Genetic-based optimization algorithm for aligning two partially overlapped scans. A novel fitness function has been proposed and experimentally evaluated. The fitness function gives the genetic algorithm several important properties, namely it does not require computing any point-to-point or feature-to-feature correspondence and is very fast, because it is based on a lookup table.

Instead of searching for corresponding point pairs and then computing the mean of the distances between them, as in other genetic algorithm's fitness functions, we evaluate the fitness directly by matching points which, after the projection on the same coordinate frame, fall in the search window around the previous scan. Hence, our approach does not yield a closest point but has a linear computational complexity $O(N)$, whereas YGM and ICP corresponding phases have a quadratic cost of $O(N^2)$. The experiments have shown that the limitations to the overall accuracy posed by the finite solution space (set by a number of bits used to code a position in the genes and by the extension of the search space) and quantization of the fitness function as defined with a lookup table are small if an appropriate size of the marked area is chosen for the problem at hand.

Let us now turn to the memory needed for the implementation of the lookup table. The lookup table is a 2D array requiring 1 bit of information per cell. For a scan matching in robotic applications with sonar and/or laser scanner, typical sensor ranges are between 5 and 50 m. With an exploration step size of several meters, a cell size of 10 cm or less should be appropriate. To cover a map of 100×100 m^2, a 1 million cells array is necessary, which only requires 125 kb of memory.

The experiments performed with the GLASM algorithm show that searching in the solution space with genetic optimization and fitness function using neither corresponding points selection nor matching error calculation, moreover based on a simple direct spatial relation, is the right approach for this type of problems.

The algorithm is robust and suitable for unstructured environments.

References

1. Besl, P., McKay, N.: A method for registration of 3d shapes. IEEE Transactions on Pattern Analysis and Machine Intelligence 14(2), 239–256 (1992)
2. Yamany, S., Ahmed, M., Farag, A.: A new genetic based technique for matching 3d curves and surfaces. Pattern Recognition 32, 1817–1820 (1999)
3. Lu, F., Milios, E.: Robot pose estimation in unknown environments by matching 2d range scans. Journal of Intelligent and Robotic Systems 18, 249–275 (1997)
4. Mumolo, E., Lenac, K., Nolich, M.: Spatial map building using fast texture analysis of rotating sonar sensor data for mobile robots. International Journal of Pattern Recognition and Artificial Intelligence 19(1), 1–20 (2005)

5. Lingemann, K., Surmann, H., Nuchter, A., Hertzberg, J.: Indoor and outdoor localization for fast mobile robots. In: Proc. of the 2004 IEEE/RSJ Int. Conference on Intelligent Robots and Systems (IROS), vol. 3, pp. 2185–2190 (2004)
6. Cox, I.: BLANCHE: An experiment in guidance and navigation of an autonomous robot vehicle. IEEE Transactions on Robotics and Automation 7(2), 193–204 (1991)
7. Biber, P., Strasser, W.: The normal distributions transform: A new approach to laser scan matching. In: Proc. of the 2003 IEEE/RSJ Int. Conference on Intelligent Robots and Systems (IROS), vol. 3, pp. 2743–2748 (2003)
8. Chen, Y., Medioni, G.: Object modelling by registration of multiple range images. Image and Vision Computing 10, 145–155 (1992)
9. Zhang, Z.: Iterative point matching for registration of free-form curves. Technical report, INRIA Tech. Rep. 1658 (1992)
10. Pfister, S., Kriechbaum, K., Roumeliotis, S., Burdick, J.: Weighted range sensor matching algorithms for mobile robot displacement estimation. In: Proc. of the 2002 IEEE Int. Conference on Robotics and Automation (ICRA), pp. 1667–1674 (2002)
11. Hahnel, D., Schulz, D., Burgard, W.: Map building with mobile robot in populated environment. In: Proc. of the 2002 IEEE/RSJ Int. Conference on Intelligent Robots and Systems (IROS), vol. 1, pp. 496–501 (2002)
12. Weiss, G., Wetzler, G., von Puttkamer, E.: Keeping track of position and orientation of moving indoor systems by correlation of range-finder scans. In: Proc. of the 1994 IEEE/RSJ Int. Conference on Intelligent Robots and Systems (IROS), vol. 1, pp. 595–601 (1994)
13. Konolige, K., Chou, K.: Markov localization using correlation. In: Proc. of the Sixteenth International Joint Conference on Artificial Intelligence (IJCAI), pp. 1154–1159 (1999)
14. Thrun, S., Burgard, W., Fox, D.: A real-time algorithm for mobile robot mapping with applications to multi-robot and 3d mapping. In: Proc. of the 2000 IEEE Int. Conference on Robotics and Automation (ICRA), pp. 321–328 (2000)
15. Neugebauer, P.: Geometrical cloning of 3d objects via simultaneous registration of multiple range images. In: IEEE International Conference on Shape Modeling and Applications, pp. 130–139 (1997)
16. Fitzgibbon, A.: Robust registration of 2d and 3d point sets. In: Conference on British Machine Vision, pp. 411–420 (2001)
17. Rusinkiewicz, S., Levoy, M.: Efficient variants of the icp algorithm. In: 3rd International Conference on 3D Digital Imaging and Modelling, pp. 145–152 (2001)
18. Agrawal, A., Ansari, N., Hou, E.: Evolutionary programming for fast and robust point pattern matching. In: Proc. of the 1994 IEEE/RSJ Int. Conference on Intelligent Robots and Systems (IROS), vol. 3, pp. 1777–1782 (1994)
19. Blais, G., Levine, M.: Registering multiview range data to create 3d computer objects. IEEE Transaction on Pattern Analysis and Machine Intelligence 17, 820–824 (1995)
20. Martínez, J.: Mobile robot self-localization by matching successive laser scans via genetic algorithms. In: 5th IFAC International Symposium on Intelligent Components and Instruments for Control Applications, pp. 1–6 (2003)
21. Martinez, J., Gozales, J., Morales, J., Mandow, A., Garcia-Cerezo, A.: Mobile robot motion estimation by 2d scan matching with genetic and iterative closest point algorithms. Journal of Field Robotics 23(1), 21–34 (2006)
22. Silva, L., Bellon, O.P., Boyer, K.: Enhanced, robust genetic algoritm for multiview range image registration. In: 4th Int. Conference on 3D digital imaging and modeling, pp. 268–275 (2003)

23. Lomonosov, E., Chetverikov, D., Ekart, A.: Pre-registration of arbitrarily oriented 3d surfaces using a genetic algorithm. Pattern Recognition Letters, Special Issue on Evolutionary Computer Vision and Image Understanding 27, 1201–1208 (2006)
24. Lenac, K., Mumolo, E., Nolich, M.: Fast genetic scan matching using corresponding point measurements in mobile robotics. In: EVOIASP 2007 European Workshop on Evolutionary Computation in Image Analysis and Signal Processing, pp. 375–382. Springer, Heidelberg (2007)

Distributed Differential Evolution for the Registration of Satellite and Multimodal Medical Imagery

Ivanoe De Falco, Antonio Della Cioppa, Domenico Maisto,
Umberto Scafuri, and Ernesto Tarantino

Abstract. In this chapter, a parallel software system based on differential evolution for the registration of images is designed, implemented and tested on a set of 2-D images in two different fields, i.e. remote sensing and medicine. Two different problems, i.e. mosaicking and changes in time, are faced in the former application field. Registration is carried out by finding the most suitable affine transformation in terms of maximization of the mutual information between the first image and the transformation of the second one, without any need for setting control points. A coarse-grained distributed version is implemented on a cluster of personal computers.

1 Introduction

Registration is a fundamental task in image processing. During the years, several techniques have been developed for various applications, resulting in several methods [1, 2]. Typically, registration is a crucial step in the fields of computer vision [3–12], medical imaging [13–20] and remote sensing [21–30].

Image registration methods proposed in literature consist of the following four steplike components [1, 2]:

- *Feature detection*. Prominent and distinctive objects (closed-boundary regions, edges, contours, corners, line intersections, etc.) are manually or, preferably, automatically detected. For further processing, these features can be represented by

Ivanoe De Falco, Domenico Maisto, Umberto Scafuri, and Ernesto Tarantino
ICAR–CNR, Via P. Castellino 111, 80131 Naples, Italy
e-mail: `{ivanoe.defalco,domenico.maisto}@na.icar.cnr.it`,
`{umberto.scafuri,ernesto.tarantino}@na.icar.cnr.it`

Antonio Della Cioppa
Natural Computation Lab, DIIIE, University of Salerno,
Via Ponte don Melillo 1, 84084 Fisciano (SA), Italy
e-mail: `adellacioppa@unisa.it`

their point representatives (centers of gravity, line endings and distinctive points), called control points.

- *Feature matching*. The correspondence between the features detected in the sensed image and those identified in the reference image is verified. For this, feature descriptors and similarity measures along with spatial relationships among the features are used.
- *Transform model estimation*. The type and parameters of the so-called mapping functions, aligning the sensed image with the reference image, are estimated. The parameters of the mapping functions are calculated by means of the established feature correspondence. According to the transformation model used, mapping functions can be classified into linear transformations, which are a combination of translation, rotation, global scaling, shear and perspective components, and elastic or "nonrigid" transformations, which allow local warping of image features.
- *Image resampling and transformation*. The sensed image is transformed by means of the mapping functions. Image values in non-integer coordinates are computed by the appropriate interpolation technique.

Among the transformation methods, the one based on the use of an affine transformation [31] to "align" at best the two images to be registered appears of interest in many fields of application. Then, the problem becomes that of finding the best among all the possible transformations, each of which is represented by a set of real parameters. An exhaustive search becomes impracticable in case of diverse degrees of freedom of the transformation, and thus, heuristic optimization algorithms are helpful. As Evolutionary Algorithms (EAs) [32–35] are successfully applied to face several multivariable optimization tasks, they have also been introduced in Image Registration, particularly in the medical [36–42] and in the remote sensing [43–48] areas.

The goal of this chapter is to design and implement an evolutionary system for the registration of images by using the affine transformation model.

DE [34, 49] is a version of an EA, which has proven to be fast and reliable in many applications [50–52]. Therefore, we have implemented a DE algorithm to find the optimal combination of the parameter values involved in the affine transformation. There exist in literature several approaches based on either explicitly providing a set of control points [21, 48, 53–55] (including DE [56]) or automatically extracting them from the image [30, 57, 58]. In contrast to those approaches, here, we wish to examine the ability of DE to perform automatic image registration without making any use of control points. This evolutionary system will be tested by means of a set of 2-D satellite images.

The chapter is structured as follows: Section 2 describes the image registration problem and defines the affine transformation and the mutual information. Section 3 contains a DE basic scheme and illustrates the application of our system to the registration task. Section 4 reports on the three problems faced, the first two of which are typical of remote sensing, i.e. mosaicking and changes in time, while the third is the classical medical problem of alignment of patients' images taken with different methods; the same section shows the results achieved by our tool. Finally, Sect. 5 contains conclusions and future works.

2 Image Registration

Registration is often necessary for integrating information taken from different sensors (*multimodal analysis*) or finding changes in images acquired under diverse viewing angles (*multiview analysis*) or disparate times (*multitemporal analysis*). Depending on the application, the goals of registering images may be quite different. In remote sensing, two problems are usually faced, i.e. *Mosaicking* and *Change Discovery*. The former is an example of *multiview analysis* and deals with spatially aligning two images of neighbouring areas taken at the same time so as to obtain a larger view of the surveyed scene, whereas the latter, representing a *multitemporal analysis* application, consists of first aligning two images of about the same area but acquired at different times, and then pointing out the changes in that area within the difference timespan. When dealing with medical images, instead, the typical task is to align at best two images from a same patient, which are the results of two different examinations such as a Magnetic Resonance (MR) and a Computer-aided Tomography (CT) or a Single Photon Emission Computed Tomography (SPECT). Therefore, this latter is a typical application of *multimodal analysis*.

In all cases, two choices must be made to carry out image registration. The first choice involves the kind of geometric transformation to be considered to find correlations between the given images, while the second one concerns the measure of match (MOM), i.e. the feature on the value of which the goodness of the registration is evaluated. Once these choices are made, the MOM can be maximized by using suitable optimization algorithms.

Affine Transformation. The most frequently used transformation model in registration is the affine transformation. It is sufficiently general and can handle rotations, translations, scaling and shearing. It can be represented in the most general 3-D case as $\mathbf{x}' = A \cdot \mathbf{x} + \mathbf{b}$, where A is a 3×3 square matrix accounting for rotations, scalings and shears, while $\mathbf{x}$, $\mathbf{x}'$ and $\mathbf{b}$ are 3-D arrays representing the original positions, the transformed ones and a translation vector, respectively.

Mutual Information. The most widely employed MOM is Mutual Information (MI) [59, 60], which represents the relative entropy of the two images to be registered. In general, given two random variables Y and Z, their MI is

$$I(Y,Z) = \sum_{y,z} P_{Y,Z}(y,z) \cdot \log \frac{P_{Y,Z}(y,z)}{P_Y(y) \cdot P_Z(z)}, \tag{1}$$

where $P_Y(y)$ and $P_Z(z)$ are the marginal probability mass functions and $P_{Y,Z}(y,z)$ is the joint probability mass function. The MI registration criterion states that the image pair is geometrically aligned through a transformation $\mathbf{T}$ when $I(Z(\mathbf{x}), Y(\mathbf{T}(\mathbf{x})))$ is maximal. Thus, the aim is to maximize Eq. (1).

3 Differential Evolution

Differential Evolution is a stochastic, population-based optimization algorithm [49]. Given a maximization problem with m real parameters, DE faces it by randomly initializing a population consisting of n individuals each made up of m real values. Then, the population is updated from one generation to the next by means of many different operators. Among them, we have chosen the one referred to as *DE/rand/1/bin*. It takes into account the generic i-th individual in the current population and randomly generates three integer numbers r_1, r_2 and r_3 in $[1,n]$ differing from one another and different from i. Moreover, another integer number k in $[1,m]$ is randomly chosen. Then, starting from the i-th individual, a new trial one i' is generated whose j-th component is given by

$$x_{i',j} = x_{r_3,j} + F \cdot (x_{r_1,j} - x_{r_2,j}) \tag{2}$$

provided that either a random real number ρ in $[0.0, 1.0]$ is lower than a value CR (parameter of the algorithm, in the same range as ρ) or the position j under account is exactly k. If neither is verified, then a copy takes place: $x_{i',j} = x_{i,j}$. F is a real and constant factor in $[0.0, 1.0]$, which controls the magnitude of the differential variation $(x_{r_1,j} - x_{r_2,j})$, and is a parameter of the algorithm.

The new individual i' is compared to the i-th in current population and the fittest is copied into the new population. This scheme is repeated for each individual and for a maximum number of generations g.

3.1 DE Applied to Image Registration

Encoding. We decided to use the affine transformation model. Since the experiments reported in this chapter refer to couples of two-dimensional images, the problem consists of finding the most suitable best combination of six real-valued parameters. Therefore, any individual in the DE population is an array with six positions, $\mathbf{T} = (a_{11},\ a_{12},\ a_{21},\ a_{22},\ b_1,\ b_2)$, and, in general, each parameter can vary within a range of its own.

Fitness. Given two images, C and D, we consider their mutual information I as the fitness function so the aim of the problem becomes to find the best transformation $\mathbf{T}$ for D such that the mutual information of C and $\mathbf{T}(D)$ is maximized.

4 The Distributed DE Algorithm

Our Distributed DE (DDE) algorithm is based on the classical coarse-grained approach to EAs, widely known in literature [61]. It consists of a locally connected topology of (in our case) DE instances, where each of them is connected to μ instances only. If, for example, we arrange them as a folded torus, then each DE instance has exactly four neighbouring populations (see Fig. 1).

Moreover, every M_I generations (migration interval), neighbouring subpopulations exchange individuals. The percentage of individuals each population sends to

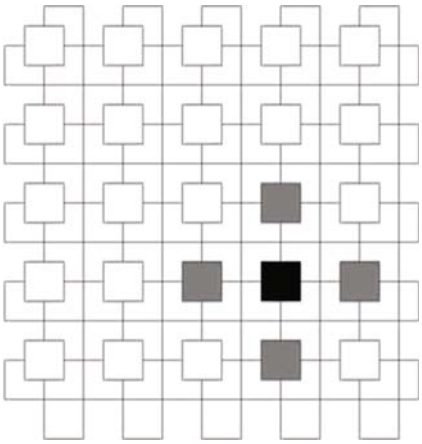


Fig. 1 The folded torus topology with $\mu = 4$

its neighbours is called migration rate (M_R). All of the chosen individuals, be their number S_I, are sent to all of the neighbours, and so each subpopulation receives a total number of $S_I \cdot \mu$ elements at each migration time.

Within this general framework, we have implemented a parallel version for DE, which consists of a set of classical DE schemes (*slaves*), running in parallel, assigned to different processors arranged in a folded torus topology, plus a *master*. The pseudo-code of any slave process is delineated below.

The master process acts as an interface to the user: it simply collects the current local best solutions of the "slave" processes and saves the best element at each generation. Furthermore, it compares this latter solution against the best found so far, and it saves the best among them and shows it to the user.

Algorithm 6. Pseudocode of the DE slave

```
begin
  randomly generate an initial population of P individuals;
  evaluate goodness of each individual;
  while termination criterion not fulfilled do
    create the new population by classical DE actions;
    evaluate goodness of individuals in the new population;
    send the current best individual to the master process;
    if time to migrate then
      send the current best individual to neighbouring populations;
      receive the best individuals from neighbouring populations;
      replace some local individuals with the received solutions;
    end if
    update variables for termination;
  end while
end
```

5 Experiments and Results

We have considered three problems. The first two, namely *Mosaicking* and *Change Discovery*, refer to remote sensing. The first accounts for the registration of two images of an area in San Francisco taken at the same time, while the second examines two images of an agricultural area near San Francisco taken at different times and looks for the changes in the area. The last one, named *Medicine*, takes into account medical imagery.

In a very preliminary set of experiments, we used a sequential DE mechanism to face the three problems. In all cases, results were encouraging; nonetheless, the computation time was quite high (tens of minutes on a 1.5 GHz Pentium 4 depending on the value of n), which led us to devote our attention to a distributed version. The DDE algorithm has been implemented in C language and communications take place via MPI. All the experiments have been effected on a Beowulf system made up of a cluster with 17 (1 master and 16 slaves) 1.5 GHz Pentium 4 nodes interconnected by a FastEthernet switch.

We have arranged the slaves in a 4×4 folded torus topology ($\mu = 4$); each DE procedure sends only its current best individual ($S_{\mathrm{I}} = 1$) and this exchange takes place every $M_{\mathrm{I}} = 5$ generations. The goodness of these choices has been confirmed by preliminary experiments. As regards the exploitation of the received solutions, these replace the worst four elements in each local population.

DE parameters have been set as follows: $n = 30$, $g = 200$, $CR = 0.5$ and $F = 0.5$. No preliminary tuning phase has been performed. It is important to remark here that, different from some papers in literature about the use of EAs to solve this task, we have decided to use quite wide ranges for each variable in the **T** solution, since we hope that evolution drive the search towards good transformations. The allowed variation ranges are shown in Table 1.

Table 1 Problem variable ranges

	a_{11}	a_{12}	a_{21}	a_{22}	b_1	b_2
min	0.500	-0.500	-0.500	0.500	-200.0	-200.0
max	1.500	0.500	0.500	1.500	200.0	200.0

For each test problem, 20 DDE executions have been carried out. The best of those runs will be described and discussed in the following in terms of image transformation achieved and evolution taken place.

5.1 *The* Mosaicking *Task*

In the first test case, we have used two images that are manually selected portions of a Landsat Thematic Mapper (TM) digital image recorded on September 7, 1984, over San Francisco Bay Area (CA, USA) (property of United States Geological Survey [62]). Those images are colour composites generated using Landsat TM spectral bands 2 (green), 4 (near-infrared) and 5 (mid-infrared) as blue, green and

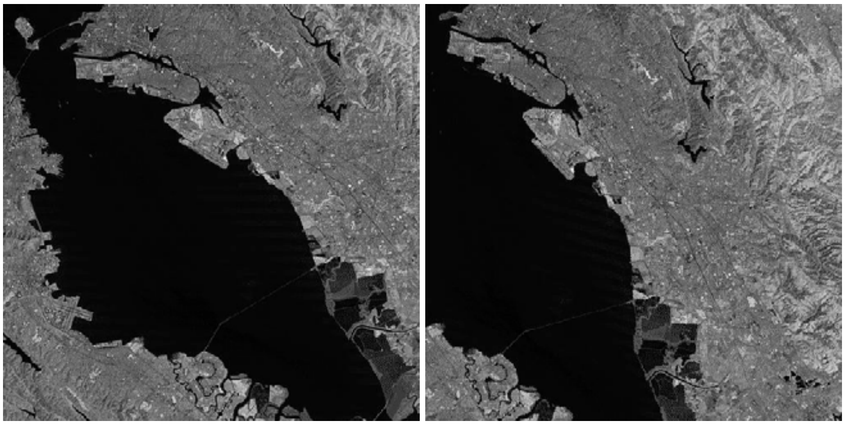


Fig. 2 The two original images for the *Mosaicking* task

red, respectively. Those images were transformed into grey monochannel images, so that each of them is 500×500 pixel large and uses 8 bits to represent each pixel. Figure 2 shows them both. Their I value is 0.1732. Figure 3 (top left) reports the fusion of the two original images. As it can be noticed, they share a common area, which should be used by the DDE algorithm to find their best registration. That is, the upper-left part of the second image overlaps the bottom-right part of the first, and a slight clockwise rotation was applied to the second image with reference to the first one. Therefore, the best affine transformation should contain a slight counterclockwise rotation and two positive shifts for both the coordinates.

The best value of I obtained in the best execution is 1.1305. The average of the best final values over the 20 runs is 1.1299 and the variance is 0.0006, the worst result being 1.1278. The best affine transformation found is $\mathbf{T} = \{0.946, -0.253, 0.253, 0.946, 42.141, 49.811\}$, which represents a counterclockwise rotation of about 15 degrees coupled with a translation in both axes. The resulting transformed image is shown in Fig. 3 (top right), while Fig. 3 (bottom left) depicts its fusion with the first image. The alignment of the two images is excellent: any detail in the first image, from streets to shorelines to bridges, is perfectly aligned with the corresponding pixels in the second one.

In Fig. 3 (bottom right), we report the evolution of the best run achieved for the *Mosaicking* task. That is, we report the best, average and worst fitness values among those sent to the master by the 16 slaves at each generation. In spite of the very relaxed parameter range allowed, the initial population achieves an improving solution with respect to the original one. From then on, the system proposes many improving affine transformations and the average, the best and the worst fitness values increase over generations until the end of the run. It must be remarked that the values for the three above-mentioned fitnesses are different from one another until generation 196, though this is quite difficult to see from the figure, especially after generation 120. Such a behaviour confirms that good solutions spread only locally among linked

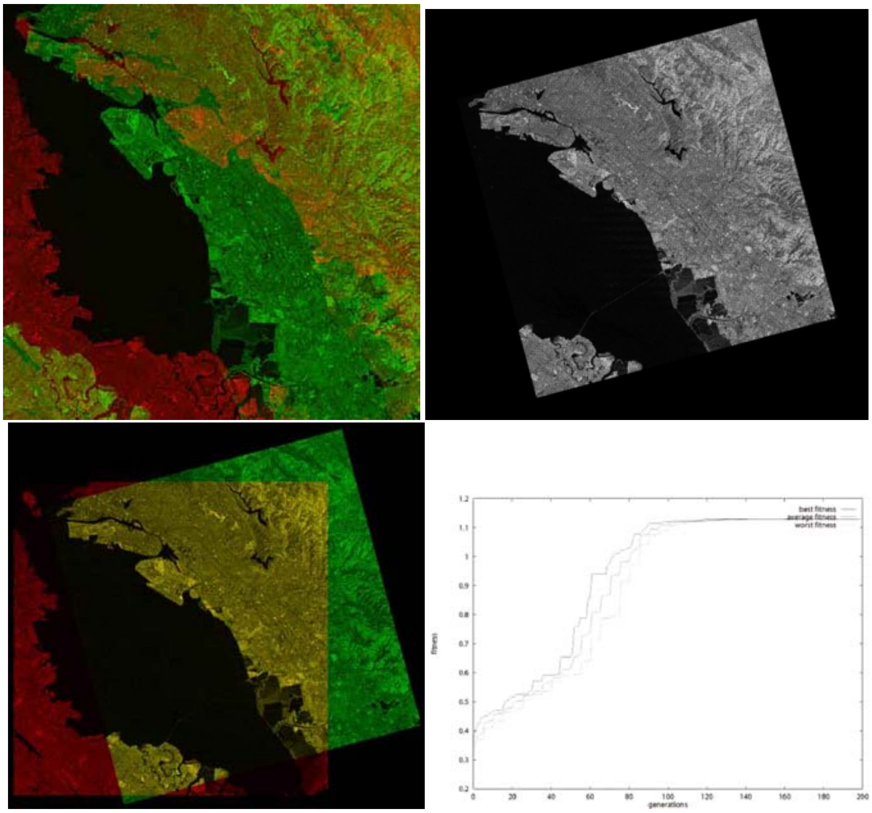


Fig. 3 (*Top Left*) The fusion of the two original images. (*Top Right*) The best transformation for the second *Mosaicking* image. (*Bottom Left*) The first image is fused with the best transformation found for the second one. (*Bottom Right*) Behaviour of fitness as a function of the number of generations for the best run

subpopulations. This local diffusion involves different processors sampling distinct search areas to avoid a premature convergence to the same suboptimal solution on all the slaves.

5.2 The Change Discovery *Task*

In the second test case, we have used two images that refer to about the same area but were taken at different times. That is, they represent an agricultural area near San Francisco (CA, USA) in 1984 and 1993, respectively (they too are property of USGS [62]). As before, the original Landsat TM images were transformed into grey monochannel images, so that each of them is 500 $\times$ 500 pixel large with an 8-bit representation for each pixel (see Fig. 4). Their I value is 0.1123. Figure 5 (top left) reports the fusion of the two original images. As it can be observed, they share a common area, which should be used by the DDE algorithm to find their

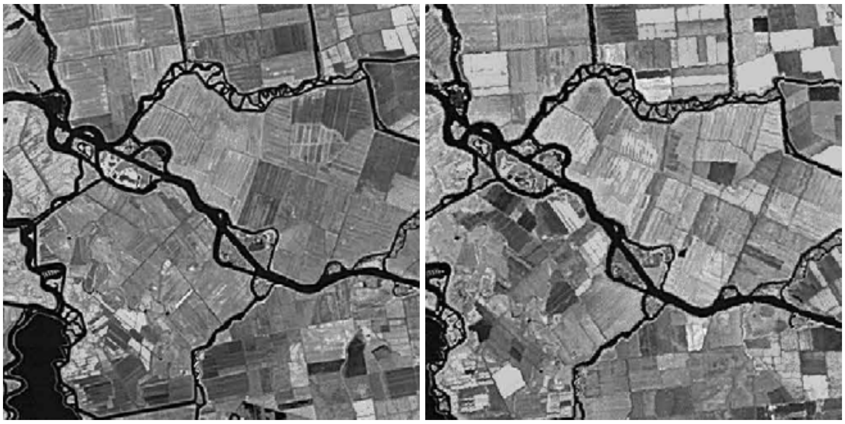


Fig. 4 The two original images for the *Change Discovery* task

best registration. That is, the right part of the first image overlaps the left part of the second, and a slight clockwise rotation took place when the second image was taken with reference to the first one. Therefore, the best affine transformation should contain a slight counterclockwise rotation and some shifts for both the coordinates.

The best value of I attained in the best execution is 0.3961. The average of the best final values over the 20 runs is 0.3959 and the variance is 0.0001, the worst result being 0.3957. The best affine transformation found is $\mathbf{T} = \{0.954, -0.083, 0.083, 0.953, 16.981, 20.282\}$, which represents a counterclockwise rotation of about 5 degrees coupled with a translation in both axes. The resulting transformed image is shown in Fig. 5 (top right), while Fig. 5 (bottom left) shows its fusion with the first image. The alignment of the two images is very good: any detail in the first image, from rivers to roads to fields, is well aligned with the corresponding pixels representing it in the second one.

Figure 5 (bottom right) presents the evolution of the best run achieved for the *Change Discovery* task. Also in this case we report the best, average and worst fitness values among those sent to the master by the 16 slaves at each generation. Also, here, despite of the very relaxed parameter range allowed, the initial population achieves an improving solution with respect to the original one, and from then on, the system proposes many improving affine transformations and the fitness values increase over generations until the end of the run. The values for the three above-mentioned fitnesses are different from one another until the end of the run, though it is quite difficult to notice from the figure, especially after generation 90. Thus considerations similar to those effected for the *Mosaicking* task about the behaviour of our tool can be made in this case too.

The computed differences between the first image and the transformed second one are shown in Fig. 6, where only the part in which the two images overlap is meaningful. In this figure the grey color refers to areas where no changes occurred, the black represents areas burnt in 1984 and recovered by 1993, whereas the white stands for areas more vegetated in 1984 than in 1993 due to differences in the

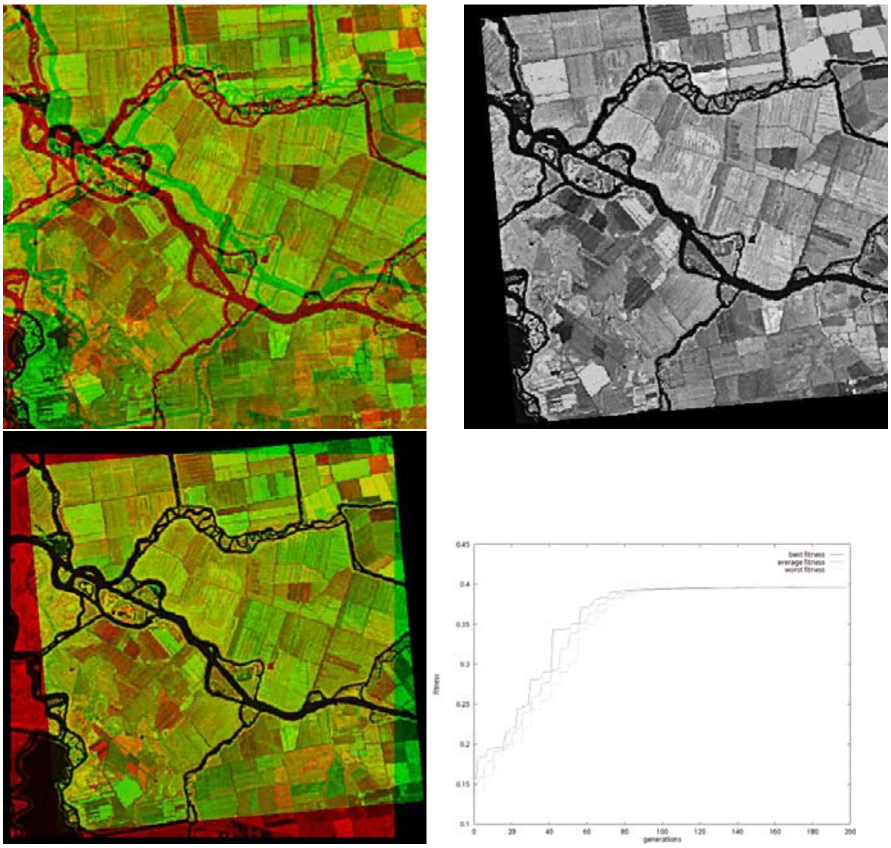


Fig. 5 (*Top Left*) The fusion of the two original images.(*Top Right*) The best transformation for the second *Change Discovery* image. (*Bottom Left*) The first image is fused with the best transformation found for the second one. (*Bottom Right*) Behavior of fitness as a function of the number of generations for the best run

amount of rainfall or the density of the vegetation. Moreover, light pixels represent areas burned in 1993 or natural landscape areas in 1984 that were converted to agricultural lands and recently tilled and finally, dark pixels stand for areas more vegetated in 1993 than in 1984.

5.3 The **Medicine** *Task*

In this test case, we have used two images that are the result of, respectively, a Photon Emission Tomography (PET) and a CT examinations taken from the abdomen of an adult male. The former was originally a 128×128 RGB colour image, while the second is 512×512 pixel large and uses 8 bits to represent each pixel. In order to achieve images of equal size, the former image has been resized up to 512×512 and has been transformed into a monochromatic image with 8 bits per pixel. Figure 7

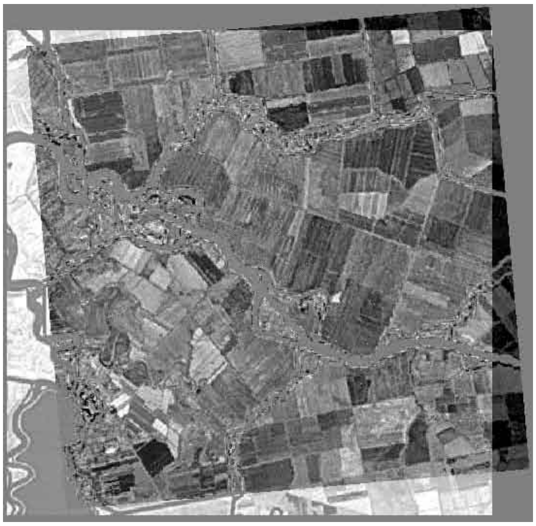


Fig. 6 Change image in an agricultural area near San Francisco within 1984 and 1993

shows them both. Their *I* value is 0.5771. It is worth noting that this problem is quite different from the previous ones: in fact, the two images contain different views of the same part of the body, and some organs are evident in the first image but are not present in the second one, and vice versa. Thus, this problem might be even more difficult from the point of view of registration. Figure 8 (top left) reports the fusion of the two original images. As it can be noticed, they are quite poorly aligned, and rotation and shifts in both dimensions should be applied to effectively register them. As an example, the top-left and bottom-right parts of the PET go beyond the limits of the CT, while the top-right and the bottom-left are too internal.

The best value of *I* obtained in the best execution is 0.6873. The average of the best final values over the 20 runs is 0.6870 and the variance is 0.0002,

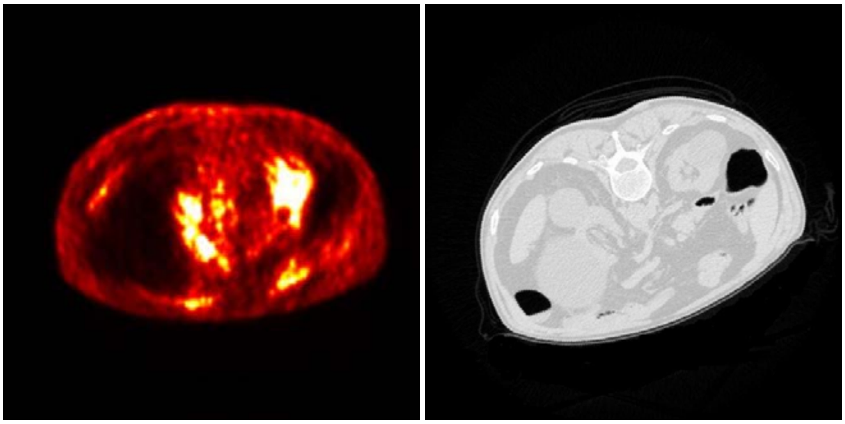


Fig. 7 The two original images for the *Medicine* task: (*left*) the PET and (*right*) the CT

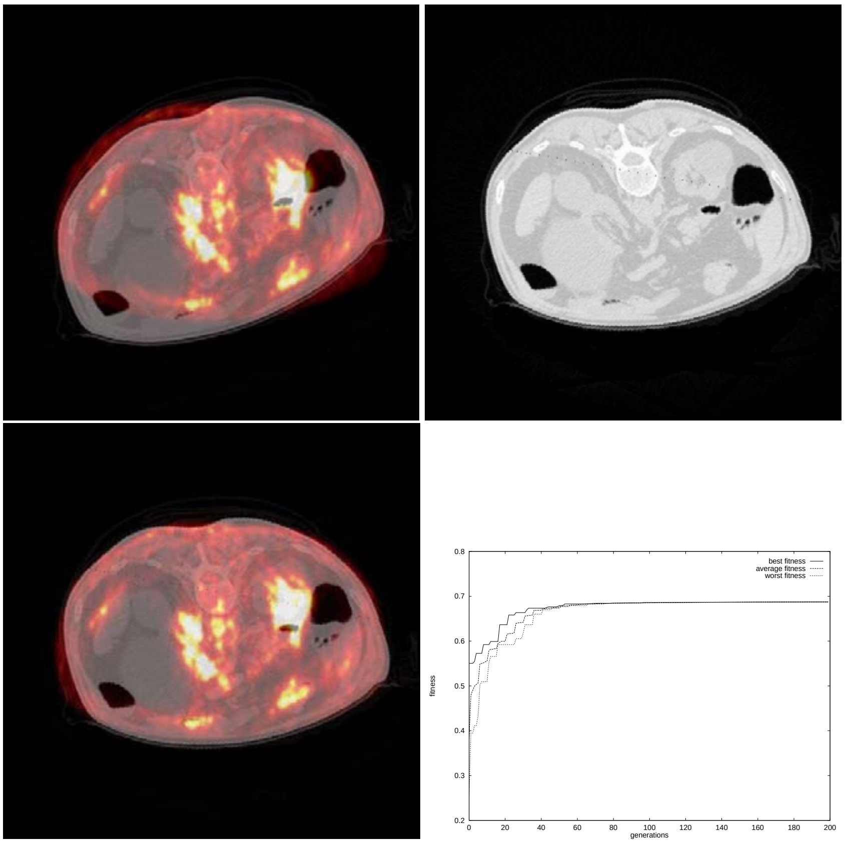


Fig. 8 (*Top Left*) The fusion of the two original images. (*Top Right*) The best transformation for the second *Medicine* image. (*Bottom Left*) The first image is fused with the best transformation found for the second one. (*Bottom Right*) Behaviour of fitness as a function of the number of generations for the best run

the worst result being 0.6865. The best affine transformation found is $\mathbf{T} = \{0.982, 0.182, -0.002, 1.000, -48.279, 8.518\}$, which represents a clockwise rotation of about 18 degrees coupled with a translation in both axes, leftbound as regards the *x* axis and downbound as for the *y* axis. The resulting transformed image is shown in Fig. 8 (top right), while Fig. 8 (bottom left) depicts its fusion with the first image. The alignment of the two images is very improved with reference to the two original images: any organ which can be seen in one image but not in the other is well aligned with the others resulting from the other examination. For example, the organs resulting from the PET examination are almost perfectly contained within the boundary of the body as provided by the CT, the only exception being the very leftmost part: had the shift to the left been slightly larger, the registration would have been excellent. To conclude, the whole part of body under investigation is well composed.

In Fig. 8 (bottom right), we report the evolution of the best run achieved for the *Medicine* task. Also for this task we report the best, average and worst fitness values among those sent to the master by the 16 slaves at each generation. The graph shows that for the present problem, the DDE takes 8 generations to find an improving solution with respect to the original one. As in the previous cases, from then on, the system proposes many improving affine transformations and the average, the best and the worst fitness values increase over generations until the end of the run. It must be remarked that the values for the three above-mentioned fitnesses are different from one another in the whole run, though this is quite difficult to see from the figure, especially after generation 70. This confirms, as before, that good solutions spread only locally among linked subpopulations without causing premature convergence to the same suboptimal solution on all the slaves.

Speedup. In all the faced problems, we need to compare the time spent by DDE (t_{DDE}, of about 7 min and 35 s) against the time t_{seq} of a sequential version using a population equal to the total number of individuals in the distributed algorithm, i.e. $16 \times 30 = 480$. This latter is of about 1 h and 51 min, which results in a speedup $s = t_{\mathrm{seq}}/t_{\mathrm{DDE}} = 6660/455 = 14.63$ and the related efficiency is $\varepsilon = s/17 = 0.86$.

6 Conclusions and Future Works

In this chapter, a distributed version of the DE strategy has been coupled with affine transformation and Mutual Information maximization to perform image registration in two fields, i.e. remote sensing and medicine, without any need of control points. A cluster of 17 personal computers has been used. The results seem to imply that this approach is promising, yet there is plenty of work to do. Therefore, future works shall aim to shed light on the effectiveness of our system in this field, and on its limitations as well. A wide tuning phase shall be carried out to investigate if some DE parameter settings are, on average, more useful than others and to analyze the influence of the parameters on performance. This phase shall take into account lots of image couples taken from different fields. A comparison must be carried out against the results achieved by other image registration methods to examine the effectiveness of our proposed approach.

References

1. Brown, L.G.: A survey of image registration. ACM Computing Surveys 24(4), 325–376 (1992)
2. Zitova, B., Flusser, J.: Image registration methods: a survey. Image and Vision Computing 21, 977–1000 (2003)
3. Zheng, Q., Chellappa, R.: A computational vision approach to image registration. IEEE Transactions on Image Processing 2(3), 311–326 (1993)
4. Blais, G., Levine, M.D.: Registering multiview range data to create 3d computer objects. IEEE Transactions on Pattern Analysis and Machine Intelligence 17, 820–824 (1995)

5. Dorai, C., Wang, G., Jain, A.K., Mercer, C.: Registration and integration of multiple object views for 3d model construction. IEEE Transactions on Pattern Analysis and Machine Intelligence 20(1), 83–89 (1998)
6. Tarel, J.P., Boujemaa, N.: A coarse to fine 3d registration method based on robust fuzzy clustering. Computer Vision and Image Understanding 73(1), 14–28 (1999)
7. Giachetti, A.: Matching techniques to compute image motion. Image and Vision Computing 18, 247–260 (2000)
8. Williams, J., Bennamoun, M.: Simultaneous registration of multiple corresponding point sets. Computer Vision and Image Understanding 73, 117–142 (2001)
9. Caspi, Y., Irani, M.: Aligning non–overlapping sequences. International Journal of Computer Vision 48(1), 39–51 (2002)
10. Masuda, T.: Registration and integration of multiple range images by matching signed distance fields for object shape modeling. Computer Vision and Image Understanding 87, 51–65 (2002)
11. Rohr, K., Forenefett, M., Stiehl, H.S.: Spline-based elastic image registration: Integration of landmark errors and orientation attributes. Computer Vision and Image Understanding 90, 153–168 (2003)
12. Yu, X., Sun, H.: Automatic image registration via clustering and convex hull vertices matching. In: Li, X., Wang, S., Dong, Z.Y. (eds.) ADMA 2005. LNCS, vol. 3584, pp. 439–445. Springer, Heidelberg (2005)
13. Christensen, G.E., Rabbitt, R.D., Miller, M.I.: Deformable templates using large deformation kinematics. IEEE Transactions on Image Processing 5(10), 1435–1447 (1996)
14. Maintz, J.B.A., Viergever, M.A.: A survey of medical image registration methods. Medical Image Analysis 2(1), 1–37 (1998)
15. Lester, H., Arridge, S.: A survey of hierarchical non–linear medical image registration. Pattern recognition 32(1), 129–149 (1999)
16. Hellier, P., Barillot, C., Mémin, E., Pérez, P.: Medical image registration with robust multigrid techniques. In: Taylor, C., Colchester, A. (eds.) MICCAI 1999. LNCS, vol. 1679, pp. 680–687. Springer, Heidelberg (1999)
17. Hill, D.L.G., Batchelor, P.G., Holden, M., Hawkes, D.J.: Image registration methods: a survey. Physics in Medicine and Biology 46, 1–45 (2001)
18. Makela, T., Clarysse, P., Sipila, O., Pauna, N., Quoc Cuong Pham Katila, T., Magnin, I.E.: A review of cardiac image registration methods. IEEE Transactions on Medical Imaging 21(9), 1011–1021 (2002)
19. Sakellaropoulos, G.C., Kagadis, G.C., Karystianos, C., Karnabatidis, D., Constatoyannis, C., Nikoforidis, G.C.: An experimental environment for the production, exchange and discussion of fused radiology images, for the management of patients with residual brain tumour disease. Medical Informatics & the Internet in Medicine 28(2), 135–146 (2003)
20. Guetter, C., Xu, C., Sauer, F., Hornegger, J.: Learning based non-rigid multi-modal image registration using kullback-leibler divergence. In: 8th International Conference on Medical Image Computing and Computer-Assisted Intervention, Palm Springs, CA, USA, pp. 255–262 (2005)
21. Ton, J., Jain, A.K.: Registering landsat images by point matching. IEEE Transactions on Geoscience and Remote Sensing 27(5), 642–651 (1989)
22. Fonseca, L.M.G., Manjunath, B.S.: Registration techniques for multisensor remotely sensed imagery. Photogrammetric Engineering & Remote Sensing 62(9), 1049–1056 (1996)

23. LeMoigne, J.: Towards an intercomparison of automated registration algorithms for multiple source remote sensing data. In: Image Registration Workshop, NASA GSFC, MD, USA, pp. 307–316 (1997)
24. LeMoigne, J.: First evaluation of automatic image registration methods. In: International Geoscience and Remote Sensing Symposium, Seattle, Washington, USA, pp. 315–317 (1998)
25. Lee, C., Bethel, J.: Georegistration of airborne hyperspectral image data. IEEE Transactions on Geoscience and Remote Sensing 39(7), 1347–1351 (2001)
26. LeMoigne, J., Campbell, W.J., Cromp, R.F.: An automated parallel image registration technique based on the correlation of wavelet features. IEEE Transactions on Geoscience and Remote Sensing 40(8), 1849–1864 (2002)
27. Mahdi, H., Farag, A.A.: Image registration in multispectral data sets. In: International Conference on Image Processing, Rochester, New York, USA, vol. 2, pp. 369–372 (2002)
28. Cole-Rhodes, A., Johnson, K., LeMoigne, J., Zavorin, I.: Multiresolution registration of remote sensing imagery by optimization of mutual information using a stochastic gradient. IEEE Transactions on Image Processing 12(12), 1495–1511 (2003)
29. Chen, H., Varshney, P.K., Arora, M.K.: Mutual information based image registration for remote sensing data. International Journal of Remote Sensing 24(18), 3701–3706 (2003)
30. Bentoutou, Y., Taleb, N., Kpalma, K., Ronsin, J.: An automatic image registration for applications in remote sensing. IEEE Transactions on Geoscience and Remote Sensing 43(9), 2127–2137 (2005)
31. Hart, G.W., Levy, S., McLenaghan, R.: Geometry. In: Zwillinger, D. (ed.) CRC Standard Mathematical Tables and Formulae. CRC Press, Boca Raton (1995)
32. Goldberg, D.: Genetic Algorithms in Optimization, Search and Machine Learning. Addison-Wesley, New York (1989)
33. Eiben, A.E., Smith, J.E.: Introduction to Evolutionary Computing. Springer, Heidelberg (2003)
34. Price, K., Storn, R.: Differential evolution. Dr. Dobb's Journal 22(4), 18–24 (1997)
35. Eberhart, R., Shi, Y.: Computational Intelligence: Concepts to Implementations. Morgan Kaufmann, San Francisco (2003)
36. Jacq, J., Roux, C.: Registration of non–segmented images using a genetic algorithm. In: Ayache, N. (ed.) CVRMed 1995. LNCS, vol. 905, pp. 205–211. Springer, Heidelberg (1995)
37. Chow, C.K., Tsui, H.T., Lee, T., Lau, T.K.: Medical image registration and model construction using genetic algorithms. In: International Workshop on Medical Imaging and Augmented Reality (MIAR 2001), pp. 174–179. IEEE Computer Society, Los Alamitos (2001)
38. Dony, R., Xu, X.: Differential evolution with powell's direction set method in medical image registration. In: IEEE International Symposium on Biomedical Imaging: From Nano to Macro, Arlington, VA, USA, pp. 732–735. IEEE Computer Society, Los Alamitos (2004)
39. Draa, A., Batouche, M., Talbi, H.: A quantum-inspired differential evolution algorithm for rigid image registration. In: Campilho, A.C., Kamel, M.S. (eds.) ICIAR 2004. LNCS, vol. 3211, pp. 147–154. Springer, Heidelberg (2004)
40. Talbi, H., Batouche, M.C.: Particle swarm optimization for image registration. In: IEEE International Conference on Information and Communication Technologies: From Theory to Applications, vol. 3, pp. 397–398. IEEE Computer Society, Los Alamitos (2004)

41. Salomon, M., Perrin, G.R., Heitz, F., Armspach, J.P.: Parallel differential evolution: Application to 3d medical image registration. In: Differential Evolution: A Practical Approach to Global Optimization. Natural Computing Series, pp. 393–411. Springer, Heidelberg (2005)
42. Telenczuk, B., Ledesma-Carbayo, M.J., Velazquez-Muriel, J.A., Sorzano, C.O.S., Carazo, J.M., Santos, A.: Molecular image registration using mutual information and differential evolution optimization. In: IEEE International Symposium on Biomedical Imaging: From Nano to Macro, Arlington, VA, USA. IEEE Computer Society, Los Alamitos (2006)
43. Fitzpatrick, J., Grefenstette, J., Gucht, D.: Image registration by genetic search. In: IEEE SoutheastCon Conference, pp. 460–464. IEEE Computer Society, Los Alamitos (1984)
44. Dasgupta, D., McGregor, D.R.: Digital image registration using structured genetic algorithms. In: SPIE The International Society for Optical Engineering, vol. 1776, pp. 226–234 (1992)
45. Turton, B., Arslan, T., Horrocks, D.: A hardware architecture for a parallel genetic algorithm for image registration. In: IEEE Colloquium on Genetic Algorithm in Image Processing and Vision, pp. 111–116. IEEE Computer Society Press, Los Alamitos (1994)
46. Ou, G., Chen, H., Wang, W.: Real–time image registration based on genetic algorithms. In: First International Conference on Real Time Imaging, pp. 172–176. IEEE Computer Society Press, Los Alamitos (1996)
47. Chalermwat, P., El-Ghazawi, T.A.: Multi-resolution image registration using genetics. In: International Conference on Image Processing, vol. 2, pp. 452–456 (1999)
48. Kim, T., Im, Y.: Automatic satellite image registration by combination of stereo matching and random sample consensus. IEEE Transactions on Geoscience and Remote Sensing 41(5), 1111–1117 (2003)
49. Storn, R., Price, K.: Differential evolution - a simple and efficient heuristic for global optimization over continuous spaces. Journal of Global Optimization 11(4), 341–359 (1997)
50. Price, K., Storn, R., Lampinen, J.: Differential Evolution: A Practical Approach to Global Optimization. Natural Computing Series. Springer, Heidelberg (2005)
51. Omran, M., Engelbrecht, A.P., Salman, A.: Differential evolution methods for unsupervised image classification. In: IEEE Congress on Evolutionary Computation, vol. 2, pp. 966–973. IEEE Press, Piscataway (2005)
52. De Falco, I., Della Cioppa, A., Tarantino, E.: Automatic classification of handsegmented image parts using differential evolution. In: Rothlauf, F., Branke, J., Cagnoni, S., Costa, E., Cotta, C., Drechsler, R., Lutton, E., Machado, P., Moore, J.H., Romero, J., Smith, G.D., Squillero, G., Takagi, H. (eds.) EvoWorkshops 2006. LNCS, vol. 3907, pp. 403–414. Springer, Heidelberg (2006)
53. Dai, X., Khorram, S.: A feature-based image registration algorithm using improved chain–code representation combined with invariant moments. IEEE Transactions on Geoscience and Remote Sensing 37(5), 2351–2362 (1999)
54. Rohr, K.: Landmark–based Image Analysis: Using Geometry and Intensity Models. Computational Imaging and Vision Series, vol. 21. Kluwer Academic Publisher, Dordrecht (2001)
55. Lim, Y., Kim, M., Kim, T., Cho, S.: Automatic precision correction of satellite images using the gcp chips of lower resolution. In: IEEE International Geoscience and Remote Sensing Symposium, Anchorage, Alaska, USA, vol. 2, pp. 1394–1397 (2004)

56. Thomas, P., Vernon, D.: Image registration by differential evolution. In: Irish Machine Vision and Image Processing Conference, Magee College, University of Ulster, Ireland, pp. 221–225 (1997)
57. Wen-Hao, W., Yung-Chang, C.: Image registration by control points pairing using the invariant properties of line segments. Pattern Recognition Letters 18(3), 269–281 (1997)
58. Netanyahu, N.S., Le Moigne, J., Masek, J.G.: Georegistration of landsat data via robust matching of multiresolution features. IEEE Trans. on Geoscience and Remote Sensing 42, 1586–1600 (2004)
59. Maes, F., Collignon, A., Vandermeulen, D., Marchal, G., Suetens, P.: Multimodality image registration by maximization of mutual information. IEEE Transactions on Medical Imaging 16(2), 187–198 (1997)
60. Pluim, J.P.W., Maintz, A.J.B., Viergever, M.A.: Mutual–information–based registration of medical images: a survey. IEEE Transactions on Medical Imaging 22, 986–1004 (2003)
61. Cantú-Paz, E.: A summary of research on parallel genetic algorithms. Technical Report 95007, University of Illinois at Urbana–Champaign (1995)
62. `http://terraweb.wr.usgs.gov/projects/sfbay/`

Euclidean Distance Fit of Conics Using Differential Evolution

Luis G. de la Fraga, Israel Vite Silva, and Nareli Cruz-Cortés

Abstract. In this chapter, we apply the Differential Evolution (DE) algorithm to fit conic curves, ellipse, parabola and hyperbola, to a set of given points. Our proposal minimizes the sum of orthogonal Euclidean distances from the given points to the curve; this is a nonlinear problem that is usually solved by minimizing the square of the Euclidean distances, which allows the usage of the gradient and some numerical methods based on it, such as the Gauss–Newton method. The novelty of the proposed approach is that we can utilize any distance function as the objective function because we are using an Evolutionary Algorithm. For the parabola case, it is proved that the calculation of the orthogonal point to a given point is a simple problem that can be solved using a cubic equation. We also show how to combine DE with a conventional deterministic algorithm to initialize it. We present experiments that show better results than those previously reported. In addition, our solutions have a very low variance, which indicates the robustness of the approach.

Keywords: Orthogonal distance fitting, Conic fitting, Euclidean distance fit, Orthogonal contacting condition, Differential Evolution.

1 Introduction

The problem of fitting a set of points to a conic is one of the most common problems in image processing and computer vision, where, for example, the projection of a circle is an ellipse and it is necessary to recognize that ellipse. This problem has been treated intensively in the specialized literature since more than 200 years ago

Luis G. de la Fraga and Israel Vite Silva
Cinvestav, Department of Computing
Av. IPN 2508. 07360 Mexico City, México
e-mail: fraga@cs.cinvestav.mx, ivite@computacion.cs.cinvestav.mx

Nareli Cruz-Cortés
Center for Computing Research,
National Polytechnic Institute, Zacatenco 07738, Mexico City, México
e-mail: nareli@cic.ipn.mx

[1–3]. The main alternative methods for the detection and analysis of geometric features are the Hough transform and the moment method [4]. In [5], an interesting algorithm for detecting ellipses using Hough transform and DE has been presented.

A conic can be represented implicitly by the general equation $ax^2 + bxy + cy^2 + dx + ey + f = 0$. The expression $b^2 - 4ac$ is known as the discriminant for the conic general equation and its sign and value can give us which type of conic the equation represents: if $b^2 - 4ac < 0$, then the conic equation may represent an ellipse, but it also has degenerate graphs that are circles, points or imaginary solutions (there is no graph). If $b^2 - 4ac > 0$, the conic equation represents a hyperbola or it can be two intersecting lines. If the discriminant is equal to zero, the conic equation may be a parabola, but it also can represent a line, two parallel lines or it possibly has no graph. When we substitute the value of a point (x,y) in the general conic equation, the value obtained is called *algebraic distance*, and this formulation allows the most efficient algorithms, because the fitting problem is linear and can be solved in a deterministic manner with a program that solves the generalized eigenvectors problem [2, 3].

The main disadvantage of using an algebraic fitting algorithm is that the fitted conic will be distorted if the point values are not known with enough accuracy [1], and it does not have a geometric interpretation.

The best fit is obtained by using a geometric algorithm because it takes into account the real distance (not the square) between a point and the ellipse. Now, we have a non-linear problem that can be solved if it is linearized by calculating the Taylor series expansion of the distance equation, and if we have an initial solution close to the real solution. In this case, it is necessary to iterate, and we need the derivative's direction to find the desired solution (the minimum). However, the Euclidean distance uses a square root, and its derivative is not continuous in all the search space. If the square of the Euclidean distance is used, then we can calculate a derivative that is continuous in all the search space and this approximation results in an iterative algorithm such as Gauss–Newton, steepest descent, etc. [6]. The main disadvantage of using the square of the distance is that the farthest points will have more weight (the squares) than the nearest points to the curve. Other important disadvantage of these numerical methods is that it is necessary to have an initial solution near to the optimum solution to avoid getting trapped in local optima.

In this work, we used Differential Evolution (DE) [7, 8] to solve the optimization problem of fitting a conic—ellipse, hyperbola or parabola—, to a set of given points, by means of finding the minimum of the sum of the distances. DE is a relative recent evolutionary heuristic designed to optimize problems over continuous domains. Each chromosome is represented by a set of decision variables using real representation. In a previous work [9], we used a Genetic Algorithm (GA) with binary Gray coding for chromosome's representation to resolve the optimization problem. In this work, we use DE because it shows better convergence properties than the GA and it directly uses real numbers representation. The usage of DE reduced the execution time in two orders of magnitude compared with the GA. The DE's inventors mention in [7] that it can be used to solve fitting curves' problems; to the best of our knowledge, this work is the first effort to use an evolutionary algorithm to fit conics according to the sum of distances to a set of given points.

In this work, we show that using DE for conics fitting results in the following: (1) A better algorithm than the one reported in [1]. It is better in the sense that it produces a lower error (this error is measured as the sum of the Euclidean distances). (2) It is robust, in the sense that it always gives a good result, and (3) It solves problems of ellipse fitting with constraints that cannot be solved in [1].

Of course, the disadvantage of using an evolutionary algorithm is that the algorithm takes a greater execution time compared with the traditional numerical method, and it gives a result of stochastic nature, i.e. we cannot guarantee that the algorithm will always converge to the global optimum. However, we obtained solutions with a very low variance value. The advantage of use an evolutionary algorithm is that we do not require an initial solution and its result can be used to initialize a conventional numerical method. Furthermore, we can use the real distance instead of square of the distance.

This article is organized as follows. Section 2 presents the general definition of the problem. Sections 3–5 present a new approach to calculate the orthogonal contacting point to the curve and the results of ellipse, hyperbola and parabola fitting, respectively. Section 6 presents a hybrid approach to initialize a conventional optimization algorithm with the result of DE. In Sect. 7, we discuss the results and, finally, in Sect. 8, some conclusions are drawn.

2 Problem Definition

Our work is based on the algorithm called least-squares orthogonal distances fitting (LSODF) of conics presented in [1]. Using their same notation, the problem of fitting a set of points to a conic can be established in a general form as follows:

1. As input, we have a set of m points, $\mathbf{X}_i$ $1 \leq i \leq m$, situated in a global coordinate system XY.
2. Points $\mathbf{X}'_i$ on the curve are calculated according to the distance to the respective $\mathbf{X}_i$ points, and they must be orthogonal to the conic. To solve this, the conic's reduced equation is used, centred at the origin. Then, points $\mathbf{X}_i$ must be translated to the new origin (X_c, Y_c) and rotated at an angle α to transform them to a new local coordinate system with points $\mathbf{x}_i$ that allow to apply the conic's reduced equation. The relation between both coordinate systems is $\mathbf{x} = \mathbf{R}(\mathbf{X} - \mathbf{X}_c)$, with the conic's origin $\mathbf{X}_c = (X_c, Y_c)$, and rotation matrix
$$\mathbf{R} = \begin{bmatrix} \cos\alpha & \sin\alpha \\ -\sin\alpha & \cos\alpha \end{bmatrix} .$$
3. Calculate the conic's parameters that minimize the sum of some definition of distance from $\mathbf{X}_i$ to the respective $\mathbf{X}'_i$ over the conic (or equivalently, the sum of distances from points $\mathbf{x}_i$ to points $\mathbf{x}'_i$ can be used to avoid the calculation of the inverse transform $\mathbf{X} = \mathbf{R}^{-1}\mathbf{x} + \mathbf{X}_c$).

Step 2 is the main idea proposed in [1], and this approximation of using the conic's reduced equation avoids singular solutions of the general conic equation (points, lines or imaginary solutions). In [1], the authors use the Newton's generalized method to calculate the $\mathbf{x}'_i$ point. In this work, we used the parametric equations for each conic instead; this simplifies the task of finding only the value for one variable, the parametric one, instead of two variables for a point $\mathbf{x}'_i = (x'_i, y'_i)$. Furthermore, when we used the parametric form of the parabola's reduced equation, the task of calculating the $\mathbf{x}'_i$ point is reduced to finding the solution of a cubic equation; this will be shown in Sect. 5.

The distance measure we have used is the Euclidean distance, d, defined as

$$d_i = \sqrt{(x_i - x'_i)^2 + (y_i - y'_i)^2} \tag{1}$$

for calculating the distance from the given point $\mathbf{x}_i$ to the point on the conic's curve $\mathbf{x}'_i$, in $\mathscr{R}^2$. The square of the distance is d_i^2.

To solve the optimization problem in step 3, the LSODF algorithm in [1] uses the Gauss–Newton method to estimate conic's parameters that minimize the sum of square of the distance.

Now, in our algorithm, we propose to use the DE to estimate the conics's parameters that minimize the error E. This error is calculated as the sum of distances:

$$E = \sum_{i=1}^{m} d_i \tag{2}$$

for m points and d_i defined in (1).

The DE has two main operators: selection and recombination. These operators allow the searching direction and the possible step size dependent on the position of the selected individuals.

We used the rand/1/bin DE's implementation because it is the most robust [10] and provides the best results for different kinds of problems. The first parameter means the selection type (e.g. random, individuals are randomly selected from the population). The second parameter indicates the number of chosen pairs to perform the difference (1, a single pair) and the third parameter is the recombination type (bin, binomial recombination).

3 Ellipse Fitting

An ellipse can be represented by five parameters, $(a, b, X_c, Y_c \textit{ and } \alpha)$, where a and b are the ellipse's semimajor and semiminor axes, (X_c, Y_c) is the ellipse center and α is its rotation angle. We need to obtain these five parameters.

Each individual in DE population has those five parameters. We used a population size of 30 individuals in all experiments, inclusive of the other conics. The fitness function is calculated according to (2). The selected bounds for each variable are $1 \le a, b \le 15$, $-10 \le X_c, Y_c \le 10$ and $0 \le \alpha < \pi$.

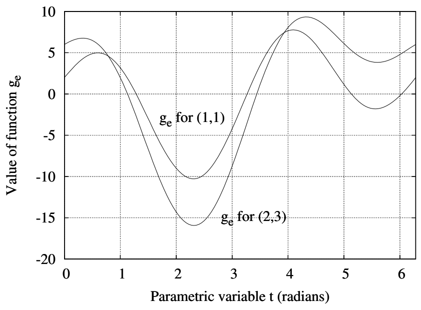


Fig. 1 Examples of the curve's form of Eq. (3), for an ellipse with $a = 4$ and $b = 2$. The values for the given point (x,y) for both curves are shown

First, in this section, we explain how to calculate the contacting point over the ellipse using its parametric equation, and later, we show the results applying DE to minimize the sum of the distances.

3.1 Orthogonal Contact Point for the Ellipse

The parametric equation of the ellipse's reduced form is $f(t) = (a\cos t, b\sin t)$, where t is the parametric variable; the derivative is $f'(t) = (-a\sin t, b\cos t)$, which also represents the ellipse's tangent vector equation, and the ellipse's normal vector is $n(t) = (b\cos t, a\sin t)$. A possible formulation to calculate the value of t corresponding to the nearest point $\mathbf{x}'$ over the ellipse, which is also normal to the point $\mathbf{x} = (x,y)$ is as follows: the equation for the line through $\mathbf{x}$ point that is normal to the ellipse is $(x - lb\cos t, y - la\sin t)$, and this line must intersect the ellipse at point $\mathbf{x}' = (a\cos t, b\sin t)$; making equal the respective terms, getting l from one equation and substituting its value in the other equation, we get:

$$g_e(t) = yb\cos t - xa\sin t + (a^2 - b^2)(\cos t)(\sin t) = 0 \, . \tag{3}$$

Roughly speaking, Eq. (3) has two solutions if point $\mathbf{x} = (x,y)$ is outside the ellipse, and it has four solutions if the given point is inside the ellipse. The curve form of (3) is very smooth, as can be seen in Fig. 1. Equation (3) was solved with Newton's method using the starting point $t_0 = \arctan[(y/a)/(x/b)]$, and this point is calculated with the atan2 programming function to allow its return value in any quadrant. The Newton's method converges in only three or four iterations because the curve is very smooth.

3.2 Results of Ellipse Fitting

In order to test our approach, we solved two problems of ellipse fitting proposed in [1]: one without constraints and another with two constraints. We also implemented in Octave [11] the algorithm used in [1] to calculate the sum of distances.

For the first problem, we used the same eight points as in [1]: $(1,7)$, $(2,6)$, $(5,8)$, $(7,7)$, $(9,5)$, $(3,7)$, $(6,2)$ and $(8,4)$. We performed 30 executions of our algorithm at several number of generations. In Fig. 2, we see the ellipse obtained at 1000

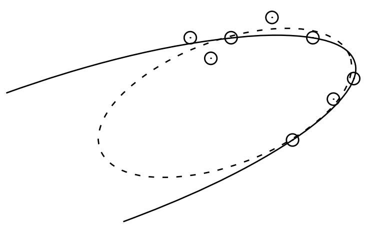


Fig. 2 Results of [1] (*dashed line*) and DE (*solid line*); the ellipse corresponding to the result at 1000 generations is shown. *Circles* mark the input points

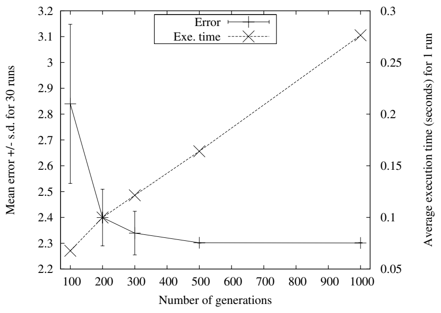


Fig. 3 The error ± standard deviation and the average execution time for 30 runs of our algorithm

generations, which is $(15.00000, 3.91504, -5.19936, 1.12509, 0.31956)$ (each value corresponds to ellipse's parameters $(a, b, X_c, Y_c \textit{ and } \alpha)$) with an error of 2.30127. We used the real distance and we have points that partially cover the ellipse, and therefore, there exists many ellipses through the given points; the best ellipse will be always in the bound for parameter a. We consider this as a difficult problem because the given points cover the ellipse only partially (see Fig. 2). The Graph in Fig. 3 shows the change of error's mean and standard deviation for the 30 executions, against the number of generations. We also show in Fig. 3 the average execution time for the 30 runs. The resulted ellipse at 500 generations has a mean error of 2.3019, with a standard deviation of ± 0.0003; that is, the resulted ellipses are equal within three decimals. At 1000 generations, the ellipses are equal within six decimals. If we calculate the error obtained in [1] $(6.51872, 3.03189, 2.69962, 3.81596, 0.35962)$, this is 2.72777, which is greater than the error obtained with our algorithm, 2.30127.

In Fig. 4, the test of our algorithm with other sets of point is shown; we ran the DE 30 times and the result that corresponds to the median error is shown.

It is interesting to comment here that if we use, in our algorithm, the square of the distance instead of the distances alone, our algorithm finds the same solution as that of the LSODF algorithm [1].

3.3 Ellipse Fitting with Constraints

The other experiment is to fit the set of points (taken from [1]) $(8,1)$, $(3,6)$, $(2,3)$, $(7,7)$, $(6,1)$, $(6,10)$ and $(4,0)$ to an ellipse with two constraints: $\alpha = \pi/2$ and $ab = 20$. Two variables are eliminated from the DE, α and b (b is calculated as $20/a$). The resulted ellipse at 100 generations is $(8.569, 2.334, 5.081, 2.123, 1.571)$ with a mean error of $3.9352 \pm$ a standard deviation of 0.00007; so our algorithm calculates the same ellipse at 100 generations within three decimal points. Our algorithm is able to reach these constraints, in contrast to the algorithm in [1], which cannot give a solution that satisfies the constraint $\alpha = \pi/2$, and of course, the error is very large

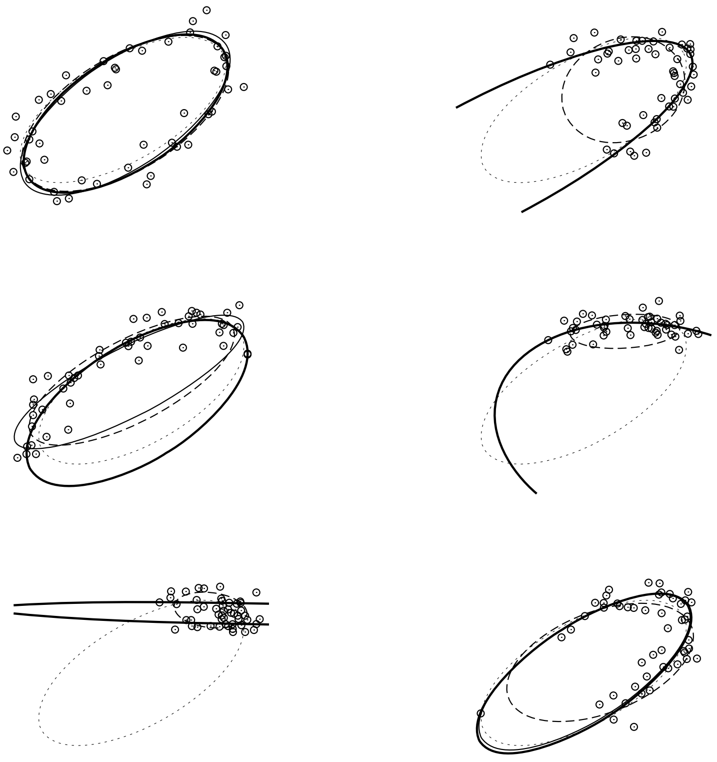


Fig. 4 We used an ellipse of axes 4.5 and 2, with the centre at $(5,4)$ and rotated 30° (*dotted line*); then, we generate 50 points and to each point position, a Gaussian noise was added to obtain a SNR of 10. We calculate three ellipses: an algebraic fitted ellipse (*dashed line*), using the LSODF algorithm [1] (*solid line*) initialized with the previous ellipse, and our DE fit (*bold line*). In some cases, the LSODF algorithm does not converge. The selected bounds were $1 \leq a, b, X_c, Y_c \leq 10$ and $0 \leq \alpha \leq \pi$

as can be seen in Table 1. There are three variables for this problem, with bounds: $1 \leq a \leq 15$ and $-10 \leq X_c, Y_c \leq 10$. The mean error measured with (2) is lower than that obtained by the algorithm in [1] because of the obvious reason that it does not satisfy the constraints. Figure 5 shows graphically the ellipse obtained by both algorithms, and Fig. 6 shows the mean error and execution time for 30 runs of our algorithm.

Table 1 Ellipse fitted with the algorithm in [1]. The constraints are $\alpha = \pi/2$, and $b = 20/a$. This result does not satisfy the second constraint. We calculate both the sum of square of the distances and the sum of distances

Variable	a	b	X_c	Y_c	$\sum d^2$	$\sum d$
Value	4.58061	4.36740	6.17691	4.46768	358.04831	49.79220

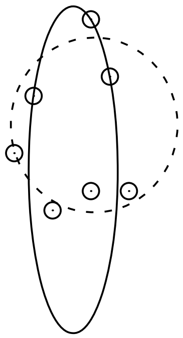


Fig. 5 Result of [1] (*dashed line*) and our DE (*continuous line*); the ellipse obtained at 100 generations is shown

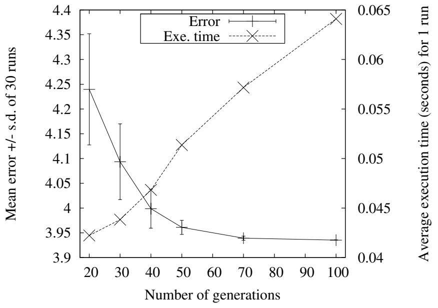


Fig. 6 The mean error ± standard deviation and the average execution time for 30 runs of our algorithm

4 Hyperbola Fitting

A hyperbola, as in the ellipse case, is represented by five parameters, (a, b, X_c, Y_c and α), where a and b are the axis lengths, (X_c, Y_c) is the coordinate centre and α is the pose angle.

First, we will show how to calculate the contacting point over the hyperbola using its parametric equation, and later the results of applying DE to minimize the sum of distances will be shown.

4.1 Orthogonal Contacting Point on Hyperbola

The hyperbola's parametric equation is $f(t) = (a\cosh t, b\sinh t)$, the tangent vector equation is equal to its derivative, $f'(t) = (a\sinh t, b\cosh t)$ and the hyperbola's normal equation is $n(t) = (-b\cosh t, a\sinh t)$. In the same way we proceeded with the ellipse, the equation of the line that contains the point $\mathbf{x}$ and is normal to the hyperbola is $(x + bl\cosh t, y - al\sinh t)$, which must intersect the hyperbola at point $(a\cosh t, b\sinh t)$; making equal the corresponding terms, getting l from one equation and resolving the other, we get:

$$g_h(t) = yb\cosh t + xa\sinh t - (a^2 + b^2)(\sinh t)(\cosh t) = 0\,. \tag{4}$$

Equation (4) was solved with Newton's method with the initial solution $t_0 = \cosh^{-1}(x/a)$; because the curve's form of (4) is also very smooth, Newton's method converges in only three iterations.

4.2 Results of Hyperbola Fitting

The same set of eight points used in [1], $(1,4)$, $(2,2)$, $(3,6)$, $(5,6)$, $(7,-1)$, $(5,0)$, $(9,8)$ and $(4,2)$, was used as input to our algorithm. We used bounds values $0.1 \le a, b \le 10$, $-10 \le X_c, Y_c \le 10$ and $-1 \le \alpha < 0$. We performed 30 executions of

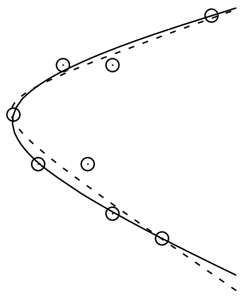


Fig. 7 Result of [1] (*dashed line*) and our DE (*continuous line*); the hyperbola obtained at 500 generations is shown

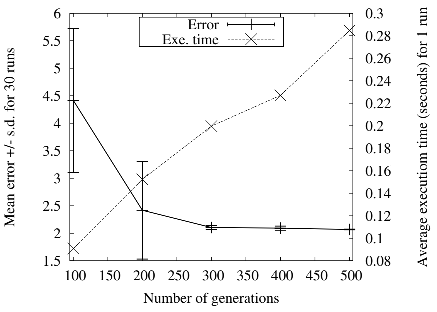


Fig. 8 The mean error ± standard deviation and the average execution time for 30 runs of our algorithm

our proposed algorithm, and the hyperbola corresponding to the median error was $(9.946, 3.301, -8.948, 4.598, -0.080)$ with an error of 2.068. The hyperbola's right part can be seen in Fig. 7. The obtained error and the execution time can be seen in Fig. 8; the error mean and its standard deviation at 500 generations were 2.073 and 0.010, respectively. The average execution time was 173% of the execution time for fitting the ellipse without constraints in Fig. 3 at 500 generations.

We performed other two experiments. With the same set of points as used in the previous experiment, we fit a hyperbola with the constraint $\alpha = 0$. We used the same program adopted for the previous experiment, but set the value of 0 for both α bounds. We got two solutions, one hyperbola is $(4.781, 1.814, -4.037, 3.398, 0.000)$, with an error of 2.865, and the other one is $(0.225, 0.112, -0.444, 3.288, 0.000)$, with an error of 2.867, both solutions are drawn in Fig. 9.

The third experiment was to fit a hyperbola to eight points [1]: $(-14, -4)$, $(-8, 7)$, $(-5, 5)$, $(-4, 2)$, $(5, 4)$, $(6, 1)$, $(9, 0)$ and $(13, -3)$. We used the same bounds

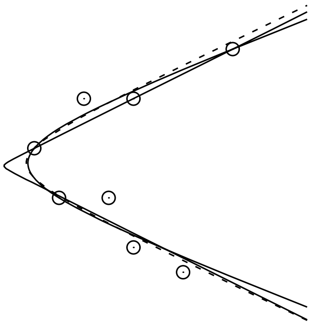


Fig. 9 Two hyperbolas obtained by fitting the same points used in Fig. 7 with constraint $\alpha = 0$. *Dashed line* shows the solution obtained in [1]

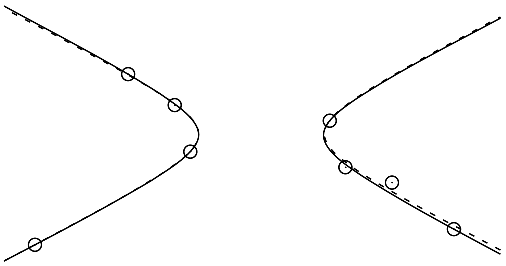


Fig. 10 Hyperbola resulted of our third experiment. *Dashed line* shows the solution obtained in [1]

adopted in the two previous tests, and the resulted hyperbola corresponding to the median error was $(4.027, 2.059, 0.570, 3.080, -0.004)$. This hyperbola can be seen in Fig. 10.

5 Parabola Fitting

A parabola is represented by four parameters, $(p, X_c, Y_c \textit{ and } \alpha)$, in a local coordinate system with focus in $(0, p)$ and directrix $x = -p$; this local coordinate system is translated to point (X_c, Y_c) and rotated at an angle α with respect to a global coordinate system XY, as is visualized in Fig. 11.

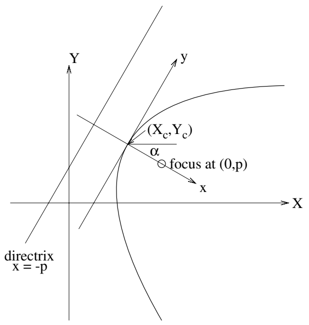


Fig. 11 A parabola with vertex at (X_c, Y_c), focus at $(0, p)$ and pose angle α

5.1 Orthogonal Contacting Point on Parabola

For the parabola, we used its parametric equation $f(t) = (pt^2, 2pt)$, its derivative is $f'(t) = (2pt, 2p)$ and the equation for its normal vector is $n(t) = (-2p, 2pt)$. As we proceeded with ellipse and hyperbola cases, the line that passes through point $\mathbf{x} = (x, y)$ and is normal to the parabola is $(x + 2lp, y - 2lpt)$; this line must intersect the parabola on point $(pt^2, 2pt)$; making the corresponding terms equal, getting l from one equation and solving the another equation, we get:

$$g_p(t) = at^3 + (2p - x)t - y = 0 \tag{5}$$

Thus, the contacting point on the parabola is $(pt^2, 2pt)$ at the value of t obtained by solving (5) for a parabola specified by the parameter p. This calculated point is then the nearest one and is orthogonal to a given point $\mathbf{x}$.

Equation (5) has a direct algebraic solution; it is not necessary to iterate to solve it, and we used the formula explained in [12] to solve it.

5.2 Results of Parabola Fitting

For DE execution, we select a population of 30 individuals. Each individual is represented by five real variables: four real variables are used for each parabola parameter $(p, X_c, Y_c \textit{ and } \alpha)$ and the fifth real variable is used to store the fitted parabola error.

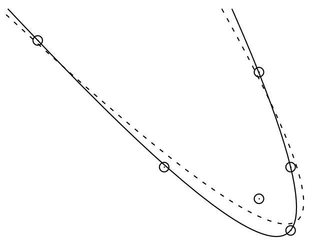


Fig. 12 Free fitting of a parabola, result of [1] (*dashed line*) and our DE (*continuous line*); the ellipse obtained at 300 generations is shown

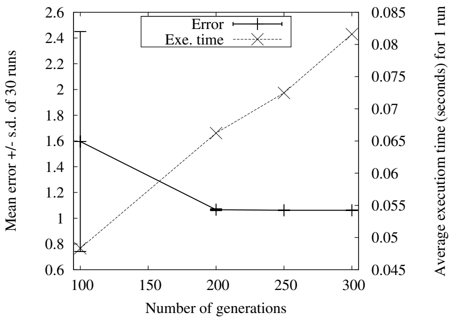


Fig. 13 The mean error $\pm$ standard deviation and the average execution time for 30 runs of our algorithm for the parabola in Fig. 12

For the six points [1], $(-7,9)$, $(-3,5)$, $(0,4)$, $(1,3)$, $(1,5)$ and $(0,8)$, we ran our algorithm 30 times using the bounds for each parameter $0.1 \leq p \leq 10$, $-10 \leq X_c, Y_c \leq 10$ and $\pi/2 \leq \alpha < \pi$. The resulted parabola at 300 generations is (0.2437, 0.8963, 2.9042, 2.1742), which is shown in Fig. 12; it has a mean error of 1.0617 and a standard deviation of 7×10^{-6}; therefore, all the obtained parabolas are the same within four decimal places. The execution time of parabola's algorithm takes approximately 67% of the time compared with the ellipse's algorithm, as can be seen from Figs. 3 and 13. This reduction in the execution time for the parabola's algorithm is because its contacting point calculation is direct, and the iterative Newton's method has not been applied to solve it. We calculate the error for the parabola obtained in [1], (0.42196/2, 1.21252, 3.31011, 2.25500), as 1.4345, which is greater than our error of 1.0617.

We performed other two experiments to fit parabolas according to the sum of orthogonal distances. For the set of five points [1], $(-1,1)$, $(2,-2)$, $(5,3)$, $(10,-4)$ and $(-4,-3)$, we used the same bounds for the variables as used before. The

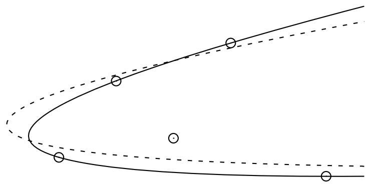


Fig. 14 Second result of free fitting of parabola. Result of [1] (*dashed line*) and our DE (*solid line*)

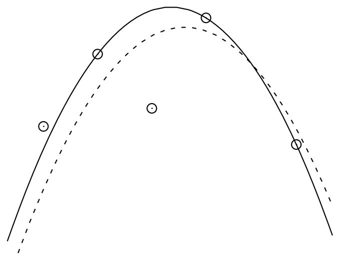


Fig. 15 Result of fitting a parabola to the same points used in Fig. 14 with the constraint $\alpha = \pi/2$. Result of [1] (*dashed line*) and our DE (*solid line*)

resulted parabola at 300 generations was $(0.2755, -5.5828, -1.9441, 0.1322)$ with an error of 1.8310 and can be seen in Fig. 14. If we calculate the error for the solution given in [1], $(0.38164/2, -6.73135, -1.30266, 0.08523)$, it is 2.7589, which is greater than 1.8310. Using the same set of points, but now with the constraint $\alpha = \pi/2$, we also get, at 300 generations, the parabola $(1.5865, 3.0577, 3.5945, -1.5708)$ with an error equal to 4.7320 (see Fig. 15). The sum of square of the distances for the parabola calculated in [1], $(3.40131/2, 3.80624, 2.49235, -1.5708)$, is 6.802079, which is greater than our error of 4.7320. We executed the same program to calculate this last parabola, but giving the same value of $\pi/2$ to both bounds of α parameter.

6 Hybrid Algorithm

Another possible application of our algorithm is for initializing a conic's fitting conventional algorithm. This idea is schematized in Fig. 16. Our DE fitting algorithm was modified to use the sum of square of the distance. After a number of generations, the best individual is selected and used as input to the LSODF algorithm. The LSODF uses the Gauss–Newton method to minimize the sum of square of the distances; if the step for each used variable is less than 1×10^{-6}, we say that the hybrid algorithm converges. If it does not, then the DE algorithm is run for another fixed number of generations, and again, the best individual is used as input to the LSODF algorithm. The whole process is repeated until the LSODF converges. In order to test this hybrid algorithm, we programmed a version for the case of ellipse fitting.

The hybrid approach in Fig. 16 was run using 10 individuals and 10 generations for one iteration of our DE algorithm, and for the application case of the ellipse shown in Fig. 5. We performed 50 runs, of which 46 got the correct result with only one iteration, and in only four of them we got the result in two iterations. We always got the correct answer that satisfies the constraint (remember that the LSODF alone was unable to solve the problem).

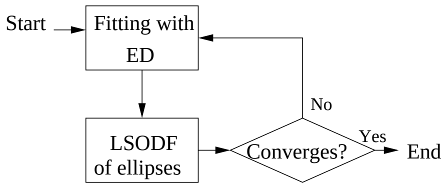


Fig. 16 A hybrid approach where our algorithm is used to initialize the LSODF algorithm

7 Discussion

The DE can also be used to fit the circle and the sphere in n dimensions to a set of given points, as is proposed in [1], but now using the sum of Euclidean distance (or other different distance measure) as the fitness function to minimize. The corresponding point on the circle or sphere is definitely described with the circle/sphere parameters and with the given point, and the algorithm to circle-sphere fitting is considered robust in [1]; because of these two facts, we did not programme those algorithms using DE. In fact, our algorithm for ellipse's fitting can be used to fit circles according to the distances to the given points if constraints $a = b$ and $\alpha = 0$ are used. However, a better circle's fitting algorithm can be programmed with an execution time as our parabola's algorithm, because it is not necessary to use an iterative algorithm to calculate the contacting point over the circle.

In a previous version of this work [9], we used a GA to perform the optimization step for ellipses. We discussed three aspects about GA versus DE: (1) the statistics for the results, (2) the number of generations and the population size and (3) the execution time.

The GA algorithm used a binary representation to encode the variables. The statistics for the results exhibited much greater variance than that using the DE that works directly with real numbers. For example, for the same ellipse fitting problem in Fig. 2, in [9], the mean error was 2.40696. Such a result was obtained with a population of 100 individuals and at 3000 generations. With DE, approximately same result was obtained with only 30 individuals and at 100 generations: a mean error of 2.3477 and standard deviation of 0.043099 was obtained, as can be seen in Fig. 3.

For the case of ellipse fitting with two constraints in Fig. 5, we reduced from 100 individuals and 1500 generations with the GA to 30 individuals and 100 generations for the DE.

The main disadvantage of GA is that its execution time was high. For one run of ellipse case, 3000 generations and population size of 100, and for eight points, the time was 27 s. The DE version, at 300 generations, 30 individuals, takes 0.1 s. A reduction in the order of two magnitudes! Execution time was measured in a iMac G5 with a 1.8 GHz processor and with Mac OS X version 10.3.9.

To know how good the values of the fitted variables are we propose to execute the DE algorithm 30 times and to calculate the covariances between each pair of variables.

8 Conclusions

We presented three new algorithms[1] that use DE to solve the problem of fitting a conic (ellipse, hyperbola and parabola) to a set of points, taking as the objective function the sum of the Euclidean distance from the given points to the curve. This problem cannot be solved by a conventional numerical method (Newton or Gauss) because the derivative of the Euclidean distance is not continuous in all the search space.

We propose here three procedures to find the orthogonal contacting point on the curve. We used the parametric form for each conic; furthermore, for the parabola case, the result is a cubic equation that has a direct solution.

The advantages of using DE are: it uses real numbers in its representation, and therefore, it give us execution times of less of a second; we do not need to initialize

[1] Source code in C is available at http://cs.cinvestav.mx/˜fraga/Softwarelibre/Conics.tar.gz

our proposed algorithm with a solution near to the global optimum, as it is required by a conventional numerical method; in all the executions, we got a reasonable solution; and it gives better results than the algorithm [1] that uses the Gauss–Newton method, according to the error measured as the sum of the Euclidean distances.

Finally, we showed that our algorithm can be used to initialize a conventional numerical algorithm, and the result of only 10 generations with 10 individuals can be used for that.

Acknowledgement. This work was partially supported by CONACyT, México, under grant 45306.

References

1. Ahn, S., Rauth, W., Warnecke, H.-J.: Least-squares orthogonal distances fitting of circle, sphere, ellipse, hyperbola, and parabola. Pattern Recognition 34(12), 2283–2303 (2001)
2. Fitzgibbon, A., Pilu, M., Fisher, R.: Direct least square fitting of ellipses. IEEE Patt. An. & Mach. Intell. 21(5) (May 1999)
3. O'Leary, P., Zsombor-Murray, P.: Direct and specific least-square fitting of hiperbolae and ellipses. Jounal of Electronic Imaging 13(3), 492–503 (2004)
4. Safaee-Rad, R., Smith, K., Benhabib, B., Tchoukanov, I.: Application of moment and fourier descriptors to the accurate estimation of elliptical shape parameters. Pattern Recognition Lett., 497–508 (1992)
5. Kasemir, K., Betzler, K.: Detecting ellipses of limited eccentricity in images with high noise levels. Image and Vision Computing 21(2), 221–227 (2003)
6. Deb, K.: Optimization for Engineering Design. Prentice-Hall, Englewood Cliffs (2002)
7. Price, K.V., Storn, R.M., Lampinen, J.A.: Differential Evolution. In: A Practical Approach to Global Optimization. Springer, Berlin (2005)
8. Price, K.V.: An introduction to differential evolution. In: Corne, D., Dorigo, M., Glover, F. (eds.) New Ideas in Optimization, pp. 79–108. McGraw-Hill, New York (1999)
9. de la Fraga, L., Vite Silva, I., Cruz-Cortés, N.: Euclidean distance fit of ellipses with a genetic algorithm. In: Giacobini, M. (ed.) EvoWorkshops 2007. LNCS, vol. 4448, pp. 359–366. Springer, Heidelberg (2007)
10. Mezura-Montes, E., Velázquez-Reyes, J., Coello Coello, C.: A comparative study of differential evolution variants for global optimization. In: Proceedings of the Genetic and Evolutionary Computation Conference (GECCO 2006), vol. 1, pp. 485–492. ACM Press, New York (2006)
11. Gnu octave, a high-level language for numerical computations, `www.octave.org`
12. Nickalls, R.: A new approach to solving the cubic: Cardan's solution revealed. The Mathematical Gazette 77, 354–359 (1993)

An Evolutionary FIR Filter Design Method

Raed Abu Zitar and Ayman Al-Dmour

Abstract. In this introductory chapter, we present an evolutionary-based technique for designing one-dimensional and two-dimensional Finite Impulse Response (FIR) filters. Typically, the required filter has a given set of specifications to be met. The specifications may include cut-off frequency, band-stop region, band-pass region and ripple factors. The evolutionary method we are using is a modified version of the Genetic Algorithm (GA), which we call Flexible Genetic Algorithm (FGA). It is an optimization algorithm with high capabilities to span the space of filter parameters. FIR filters are highly required in different applications that process signals or images. A review of the state-of-the-art of filter optimization using evolutionary techniques is presented in this chapter. The aim of this work is to simply give a basic example of how filters can be designed using evolutionary techniques. As a matter of fact, medical applications require high linearity in the filter phase function to prevent undesired distortions in the detected signals. The proposed technique is based on minimizing a cost function that uses the weighted squared difference between the optimum filter specifications and the solutions generated by the evolutionary method. Comparisons between FGA-designed filter and standard method designed filters are implemented. Testing of filters is done using different noisy artificial ECG signals and selected images. We used Hermite functions to build the artificial ECG signals.

1 Introduction

The FIR filters are particularly useful to designers who need highly linear phase systems, such as in biomedicine, where it is of paramount importance to avoid distorting the signal during filtering. To obtain a more efficient linear phase FIR filter, the linear-phase region can be confined to a portion of the pass-band and the phase left unrestricted in the stop band, where it is of less importance. This will

Raed Abu Zitar
School of Computer Science and Engineering, New York Institute of Technology, Amman, Jordan
e-mail: rzitar@philadelphia.edu.jo

Ayman Al-Dmour
Department of Information Technology, Al-Hussein Bin Talal University, Ma'an, Jordan
e-mail: d.ayman@ahu.edu.jo

allow the filter to be designed with fewer coefficients than for a fully linear phase system. To design this type of filters, both the magnitude and phase responses have to be optimized according to some sensitive objective function. Natural algorithms such as GA [1] and Simulated Annealing (SA) [2] have become popular optimization tools for performing searches of hilly and multidimensional surfaces, where traditional methods such as hill-climbing cannot perform well. Classical GA [3–5] has some limitations, and one of its main limitations is the fixed dimensionality of the solutions it can span. This is due to the fixed length strings used in the standard GA. In FGA, strings (chromosomes) have variable lengths; hence, solutions with different dimensionalities can be found.

The FGA is a general evolutionary optimization algorithm that may be used to optimize complex, multivariable, and nonlinear problems with constraints. In this chapter, the implementation of FGA in designing one-and two-dimensional FIR filters is investigated. In other words, the set of coefficients that constitutes an optimal FIR filter are sought. With this evolutionary-based design methodology, simultaneous optimization for the filter parameters is performed. Unlike the design process using standard techniques [6–8], FGA does not divide the search into stages, where each stage is optimized separately. Dividing the design process into stages only allows one to find sub-optimal solutions and not globally optimal ones. The need arises for a technique that takes into account the interrelation between different stages.

In the last two decades, there has also been a great deal of interest in designing two-dimensional (2Digital) filters due to the increasing applications in 2D digital signal processing. This interest has been boosted by the advances of very large scale integrated (VLSI) circuits, which have allowed real-time operations of 2D digital filters. 2D digital filters find applications in versatile areas such as digital processing of aerial and satellite photographs, enhancement of X-rays, computer tomography, digital television, radio astronomy, processing of geophysical data and radars, to name just a few.

The two classes of digital filters, depending on the nature of their impulse response, are the infinite-extent impulse response (IIR) filters and finite-extent impulse response (FIR) digital filters. The IIR filters are more economical; they can meet particular design specifications with a significantly smaller number of filter coefficients than do the FIR filters. However, the 2D FIR filters are widely used in the field of digital signal processing and are often preferred to the 2D IIR filters [9].

The problem of designing a digital filter is basically a problem of finding the impulse response (or transfer function) coefficients that meet the design specifications. Existing standard methods for designing 2D FIR filters are the windowing method, the frequency-sampling method, the frequency transformation method and optimal (*minimax*) methods [10]. The 2D window method is based on the same concept as its 1D counterpart. The desired filter frequency response $H_d(n_1,n_2)$ is known and the corresponding impulse response $h_d(n_1,n_2)$ is found by inverse Fourier transforming $H_d(\omega_1,\omega_2)$ .Then, this desired or ideal frequency response $H_d(n_1,n_2)$, which is of infinite extent, is truncated to a finite-extent sequence. In order to moderate the Gibbs phenomenon, this truncation is performed with a window function $\omega(n_1, n_2)$, that is, $h(n_1, n_2) = h_d(n_1, n_2)\,\omega(n_1, n_2)$.

If $h_d(n_1, n_2)$ and $\omega(n_1, n_2)$ are both symmetric with respect to the origin, then $h_d(n_1, n_2)$ will also be symmetric and the designed filter has a zero-phase frequency response $H(\omega_1,\omega_2)$. This frequency response is a smoothed version of the ideal frequency response. One-dimensional windows are often used as a basis for generating 2D windows [11–13]. There are two methods by which this is usually done. The first method is to obtain a 2D window $\omega(n_1, n_2)$ as an outer product of two 1D windows, $\omega(n_1, n_2) = \omega(n_1)\,\omega(n_2)$. The second method, proposed by Huang [13], is to obtain a 2D window $\omega(n_1,n_2)$ by sampling a circularly rotated 1D continuous window function using:

$$\omega(n_1, n_2) = \left|\omega_c\left(\sqrt{n_1^2 + n_2^2}\right)\right|. \tag{1}$$

Amongst the most popular 1D windows that can be used to obtain a 2D window are the rectangular, the Hamming and the Kaiser windows. The window method is quite general and it is not optimal. There is no control over the frequency domain specifications, and sometimes, it is necessary to design several filters to meet the design specifications. The window method is conceptually and computationally simple and it is often used. On the other hand, the frequency transformation method was originally proposed by McClellan [14] and further developed by other authors [15]. In this method, a 2D zero-phase FIR filter is designed from a 1D zero-phase FIR filter using the Fourier transformation of a finite-extent zero-phase sequence $t(n_1, n_2)$.

The optimal 2D FIR design involves optimization of the designed filter coefficients such that some function of the error between the resulting filter frequency response and the desired filter frequency response is minimized. For instance, the Chebyshev (L_{infinite}) norm below may be used:

$$E_\infty = \max_{(\omega 1.\omega 2)}\left|H(\omega_1, \omega_2) - H_d(\omega_1, \omega_2)\right|. \tag{2}$$

Note that this error function may include weighted terms.

In the next section, we will introduce how FGA can be applied in filter design. In the third section, we specifically present the 1D case, and in the fourth section, we discuss implementing the FGA for the 2D case. In the last two sections, we present the results, discussions and final conclusions.

2 Multi-objective Optimization by FGA

The basic building block in the FGA is the "string", which is a sequence of bits representing some variables of the search space. The bits form the genotype and their decoded values are the phenotype. All FGA operations are implemented over a finite population of strings.

As the search process goes on, the average fitness for the population of strings is expected to increase. Fitness is measured using some multi-objective function that is related to the criterion that needs to be optimized (in our case, the weighted Euclidean distance between the optimum desired responses and designed responses). Reproduction is done after some sorting phase, and offspring are generated using mutation and crossover. Mutation is a random flipping of bits [7]. The flow chart of Fig. 1 describes the FGA.

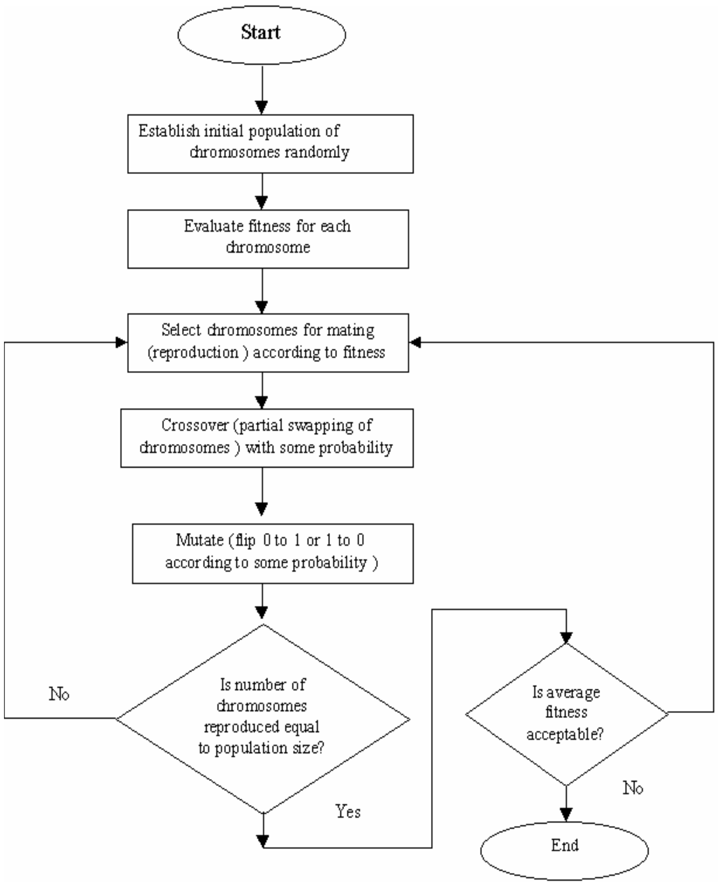

Start
Establish initial population of chromosomes randomly
Evaluate fitness for each chromosome
Select chromosomes for mating (reproduction) according to fitness
Crossover (partial swapping of chromosomes) with some probability
Mutate (flip 0 to 1 or 1 to 0 according to some probability)
Is number of chromosomes reproduced equal to population size?
No
Yes
Is average fitness acceptable?
No
End

Fig. 1 Flowchart of the FGA operations

Crossover is partial swapping of selected strings. The strings are selected according to their fitness. In our FGA, the strings do not have to posses equal lengths. That is why we call this GA "flexible GA". Since each string consists of a set of sub-strings and each sub-string is decoded to give a single filter coefficient, we will end up with a variable length set of strings. The optimum number of coefficients for our filter is unknown here. The FGA strings are "suggestions" for the number of coefficients and their values. These coefficients are used in the design of a filter and the fitness of the filter is measured according to a measure that will be shown later. The crossover is done only amongst strings with equal lengths. This is part of the restricted mating strategy [16, 17] that preserves competition only amongst similar species. However, selection of mates is done for those with high fitness and equal strings lengths; strings with different lengths are not allowed to mate regardless of their fitness.

3 The One-Dimensional Filter Design Case

The 1D FIR filters are employed in filtering problems where there is a requirement for a linear-phase characteristic within the pass band of the filter. If there is no requirement of a linear-phase characteristic, either an IIR or an FIR filter may be employed. However, as general rule, an IIR filter has lower side lobes in stop band than an FIR having the same number of parameters. For this reason, if some phase distortion is either tolerable or unimportant, an IIR filter is preferable, primarily because its implementation involves fewer parameters, it requires less memory and has lower computational complexity.

However, as mentioned earlier, biomedical signal phase distortion in pass band region is unacceptable and computational complexity should be avoided as much as possible; therefore, it is highly desirable to design an FIR filter with fewer coefficients.

An FIR filter of length M with input $x(n)$ and output $y(n)$ is described by the difference equation given below:

$$y(n) = b_o x(n) + b_1 x(n-1) + \cdots + b_{M-1} x(n-M+1) = \sum_{k=0}^{M-1} b_k x(n-k). \qquad (3)$$

Where b_k is the set of filter coefficients.

The pulse response of FIR filter is given by

$$h(n) = \sum_{k=0}^{M-1} b_k \delta(n-k) \qquad (4)$$

and the transfer function is given by

$$H(z) = \sum_{k=0}^{M-1} h(k) z^{-1} . \qquad (5)$$

An FIR filter has a linear phase if its pulse response satisfies the symmetry or anti-symmetry conditions

$$h(n) = \pm h(M-1-n), \text{n=0, 1,…, M−1} . \qquad (6)$$

When the symmetry and anti-symmetry conditions in (6) are incorporated into (5), we have

$$H(z) = z^{-(M-1)/2} \sum_{k=0}^{(M/2)-1} h(k)\left[\, z^{(M-1-2k)/2} \pm z^{-(M-1-2k)/2} \right] M\,\text{even}. \qquad (7)$$

The frequency response characteristics of linear-phase FIR filters are obtained by evaluating (7) with substitution of z=e$^{j\omega}$. This substitution yields the expression for $H(\omega)$.

When $h(n)$=h(M-1-n), $H(\omega)$ can be expressed as

$$H(\omega) = H_{\mathrm{r}}(\omega)\mathrm{e}^{-j\omega(M-1)/2}, \qquad (8)$$

where $H_{\mathrm{r}}(\omega)$ is a real function of ω and can be expressed as

$$H_{\mathrm{r}}(\omega) = h\left(\frac{M-1}{2}\right) + 2\sum_{k=0}^{(M-3)/2} h(k)\cos\omega\left(\frac{M-1}{2} - k\right) M\,\text{odd}, \qquad (9)$$

$$H_{\mathrm{r}}(\omega) = 2\sum_{k=0}^{(M/2)-1} h(k)\cos\omega\left(\frac{M-1}{2} - k\right) M\,\text{even}. \qquad (10)$$

The phase characteristic of the filter for both odd and even is

$$\varphi(\omega) = \begin{cases} -\omega\left(\dfrac{M-1}{2}\right) & \text{if } H_{\mathrm{r}}(\omega) > 0 \\ -\omega\left(\dfrac{M-1}{2}\right) \pm \pi & \text{if } H_{\mathrm{r}}(\omega) < 0 \end{cases} \qquad (11)$$

The problem of FIR filter design is simply to determine the M coefficients $h(n)$, n-1,2,…,M-1 from the specification of desired frequency response $H_{\mathrm{d}}(\omega)$ of the FIR filter. The important parameters in the specification of $H_{\mathrm{d}}(\omega)$ are as follows:

1. Pass band ripple
2. Stop band ripple
3. Pass band edge ripple
4. Stop band edge ripple

The above specification could be shortened to one parameter, which is the cut-off frequency ω_{c}. The order of the filter is also an important parameter because the optimal order is the smallest one that satisfies the above specification.

There are several basic and standard methods for designing linear phase FIR filters:

1. Windows method.
2. Frequency sampling method.
3. Optimum equi-ripple method (Chebyshev approximation).

In a very simple manner and in frequency domain, the *optimal* filter is given by

$$g(\omega) = \begin{cases} 1 & \text{where } |\omega| < \omega_c, \\ 0 & \text{elsewhere}, \end{cases} \tag{12}$$

and

$$ph(\omega) = -N\omega, \tag{13}$$

where

$g(\omega)$ is the magnitude response,
$ph(\omega)$ is the phase response,
ω_c is the normalized cut-off frequency,
ph is the phase of the ideal filter,
N is the order of the filter and
ω is the index of normalized frequency.

Optimization constraint requires that integer coefficient be used having the form:

$$a_i = \sum_{j=-nl2}^{nl2} k2^j , \tag{14}$$

where n is the length of the sub-string of the FGA, i is index of the FIR filter coefficient associated with that sub-string and k =0 or 1.

4 The Two-Dimensional Filter Design Case

The 2D finite impulse response (FIR) digital filters have impulse response $h(n_1,n_2)$ that is of a finite extent. Similar to 1D digital filters, 2D digital filters are generally specified in the frequency domain. The frequency response of a 2D digital filter is periodic with period 2π in both spatial frequencies, ω_1 and ω_2, i.e.

$$H(\omega_1,\omega_2) = H(\omega_1 + 2\pi,\omega_2) = H(\omega_1,\omega_2 + 2\pi) \text{ for all } (\omega_1,\omega_2), \tag{15}$$

where $H(\omega_1,\omega_2) = \sum_{k1=-\infty}^{\infty} \sum_{k2=-\infty}^{\infty} h(k_1,k_2)\, e^{-jk_1\omega_1} e^{-jk_2\omega_2}$

Therefore, $H(\omega_1,\omega_2)$ would be completely specified if known in the region $-\pi \le \omega_1 \le \pi$, $-\pi \le \omega_2 \le \pi$.

In many applications, for example, image processing, a zero-phase characteristic is needed. A zero-phase filter has the tendency to preserve the shape of the signal component in the pass band region of the filter. The frequency response of zero-phase 2D FIR filters is a real-valued function, i.e.

$$H(\omega_1,\omega_2) = H^*(\omega_1,\omega_2). \tag{16}$$

If in some frequency regions, $H(\omega_1,\omega_2)$ becomes negative, then a phase shift of π radians occurs. Typically, the frequency response can become negative in regions corresponding to the stop bands, and a phase of π rad in the stop bands has little significance. Provided that the impulse response $h(n_1, n_2)$ is real with a symmetric impulse response with respect to the origin of (n_1,n_2) plane, we will have:

$$h(n_1,n_2) = h(-n_1,-n_2). \tag{17}$$

In this chapter, the design of 2D rectangular-shaped zero-phase FIR filters in the frequency plane (ω_1, ω_2) is considered.

As mentioned earlier, another common method of obtaining 2D filters is to obtain them indirectly by transforming 1D prototypes low pass filters. However, the ultimate goal is to find the set of coefficients that minimizes the absolute difference between the desired filter response and the suggested filter response.

5 The FGA Implementation

The FGA algorithm is initialized by, first, setting the number of strings (population size), giving the upper and lower bounds of lengths of the strings, the number of filter coefficients (the number of sub-strings is equal to the number of filter coefficients) and providing a fitness function (objective function) to be minimized. The fitness function may be given by

$$F(\text{string}) = k_1 \,\|(g - g')\|^2 + k_2 \,\|(ph - ph')\|^2 \tag{18}$$

where:

- F is the fitness of a string in the FGA population.
- g and g' are, respectively, optimal and suggested solution vectors (the values that constitute the magnitude of the filter) defined all over the range of normalized frequency .
- ph and ph' are, respectively, optimal and suggested solution vectors (the values that constitute the phase of the filter) defined all over the range of normalized frequency.
- k_1 and k_2 are weighing constants that are used to emphasize the importance of the associated term (Note that: k_2=0 for zero-phase case i.e. the 2D case).

All the previously mentioned specifications of the FIR filter are lumped in Eq. 18. As long as this equation measures the difference between the desired magnitude and phase responses and any possible solution for an FIR filter, it can be used by our FGA to evaluate its strings.

In the 1D case, given the set of normalized specifications, and using the previously defined FIR filter design method, with k_1=1 and k_2=30 (k_2 is made larger than k_1 to make up for the fact that the magnitude response is less than or equal 1 while phase values can be much larger than 1), we implemented our "flexible" FGA. The fitness of FGA reached a steady value after around 500 iterations. By reaching a steady state, the average fitness of the population tends to settle to fixed values. This steady value represents the convergence state, and the best fitness string is the solution found by the algorithm. The number of sub-strings in the best string is equal to the number of the coefficients in the best design found by the FGA. The coefficients are taken directly from the string and used in building an FIR filter using a software toolbox available in *Matlab* 7.1. The FGA explores different possible solutions with its variable string length (i.e. different number of filter coefficients). The best fitness string achieved had a number of coefficients equal to five (i.e. the order of FIR filter was 4, see Table 1). The simulations were repeated 100 different times at different seeds for random numbers generators and the best result out of the 100 runs were taken.

Table 1 Best objective function values versus number of coefficients

Number of coefficients	Objective function values (lowest relative total error)
3	11.02
4	5.37
5	2.68
6	3.48

As a matter of fact, we understand that fixing the coefficients number to five is unfair if we are comparing with other design methods. However, we should keep in mind that the main objective of this chapter is to demonstrate how FGA can be used in FIR filter design and then how it compares to other methods.

To evaluate our work more precisely, 10 other standard FIR filter design techniques are used with the same optimum number of FGA coefficients. We did that in order to have a fair comparison between the different techniques. Some of these methods are based on windowing techniques and others are iterative optimal methods [18–20]. These techniques are probably the most popular and most frequently used. We used *Matlab* 7.1 available toolbox to build the FIR filter with the previously mentioned techniques. The same optimal filter specifications were used.

Figure 2 shows the magnitude and phase responses for the optimal filter; the FIR filter based on the Hanning window and our FGA method. We preferred not to show the responses of the other filter design types due to the lack of space. By

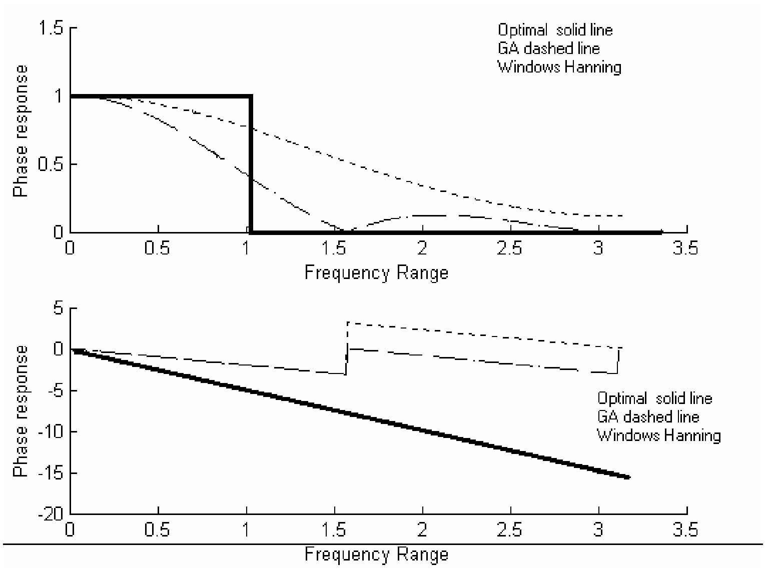


Fig. 2 Comparison between responses of optimal, FGA and windows-based Hanning FIR

Table 2 Coefficients and total error value for all types of filters relative to optimal Filter magnitude and phase responses

FIR filter design method	Filter coefficients (4th order)	Total error
Windows-based Hamming	0.0246 0.2344 0.4821 0.2344 0.0246	2.7046
Windows-based Kaiser	0.0246 0.2344 0.4821 0.2344 0.0246	2.7020
Windows-based Hanning	0.0596 0.2530 0.3747 0.2530 0.0596	2.6967
Windows-based Blackman	0.0000 0.1899 0.6203 0.1899 0.0000	2.7128
Windows-based Boxcar	0.1563 0.2210 0.2455 0.2210 0.1563	2.6909
Windows-based Bartlett	0.0000 0.2369 0.5262 0.2369 0.0000	2.7008
Optimal least square methods	0.1591 0.2259 0.2509 0.2259 0.1591	2.7079
Raised cosine	0.1591 0.2251 0.2500 0.2251 0.1591	2.6909
Optimal Chebyshev norm	0.3397 0.2049 0.2203 0.2049 0.3397	2.6909
Frequency sampling	0.0127 0.1212 0.2490 0.1212 0.0127	2.7019
FGA-based method	0.1251 0.2500 0.2501 0.2500 0.1251	2.6894

observing the magnitude and phase responses for the FGA, and comparing them with the characteristics of the optimal filter, it is not very clear that FGA filter qualitatively outperforms other filters. However, Table 2 shows a quantitative measure of "how far from the desired transfer functions" is each filter, which is based on Eq. 18 described earlier.

The next stage was testing the 1D FIR filter with different artificial ECG signals made using Hermite functions. Different levels of white noise are added to the filter input in order to test the immunity. Results will be discussed in next sections. Testing is done by passing the artificial ECG signal with or without noise through the two filters (the FGA and the Kaiser-based) and then observing the output wave that is compared to input wave.

The artificial ECG testing signals are generated using the three Hermite functions $\varnothing_0$, $\varnothing_1$ and $\varnothing_2$.

N is equal to 20.

We can make 20 different artificial ECG signals using the above template. Figure 3 consists of three parts: the upper part shows the output of the FGA filter, the middle part shows the noisy artificial ECG signal (the input signal) with signal-to-noise ratio (SNRs) of 15 and the lower part shows the output of the Kaiser filter. Kaiser filter is picked here, amongst other types of filters, because of its good output and since it represents the window-based group of filters that we would like to compare with the FGA.

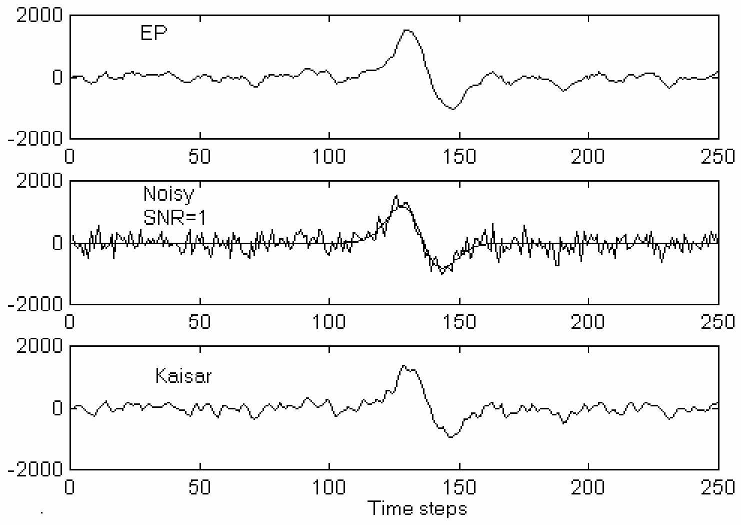


Fig. 3 (*top to bottom*) A sampled filtered signal by FGA (EP), without filtering (S/N=1) and filtered by Windows Kaiser

The total compound signal $H_j(i)$ is given by

$$H_j(i) = \begin{cases} \phi_0(i)(N-2j)/N + \phi_1(i)2j/N, & \text{for } 0 < j \le N/2 \\ \phi_0(i)(2j-N)/N + 2\phi_1(i)(N\text{-}j)/N + \phi_2(i)(j-N)/N, & \text{for } 11 \le j < N \end{cases} \tag{19}$$

For the 2D filter, we used 15×15 coefficients. This number was found optimal by the FGA for the sample image of rectangles we used. The FGA results are compared with other four standard 2D FIR design methods. The first one was the frequency sampling method which is based on a desired 2D frequency response sampled at points on the Cartesian plane. The second method was FIR filter design using the 1D window method. This method uses a 1D window specification to design the desired frequency response H_d. The third method was the 2D window specification method. It is used to design a 2D FIR filter based on the desired frequency response H_d. The fourth is the frequency transformation method. This method produces a 2D FIR filter H_d that corresponds to a 1D equivalent FIR filter, and finally, the FGA method, with which we used a 1D to 2D filter transformation routine. The FGA strings had the coefficients of a 1D filter. For the purpose of calculating the fitness, the 1D coefficients were transformed to 2D equivalents and used in building up the filter for the images that requires 2D coefficients. Equation 18 is then used for calculating the fitness of the string according to the magnitude response of those filters.

Figures 4, 5 and 6 show the image, filtered image, filter coefficients values and magnitude responses for all these methods, respectively. Typically, all these filters have zero-phase responses. It is worth mentioning that we used the Hamming window in the methods that required a window in the design, with normalized cut-off frequency of 0.5 and low-pass filtering. There is no specific reason for choosing the Hamming window except that it is popular and simple. The results obtained by the FGA were as good as any other method. The transfer function of the FGA filter was similar to those at the transformation method. This is probably due to the similarities in the two approaches.

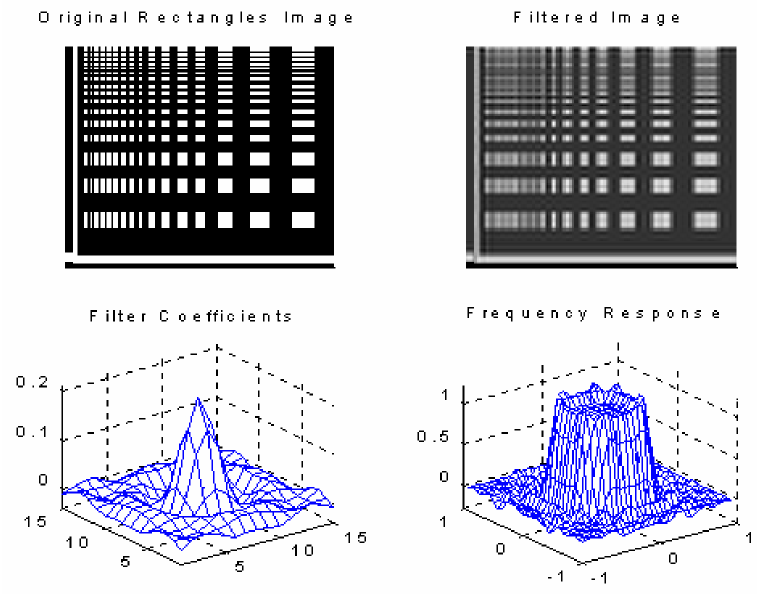


Fig. 4 The 2D FIR with frequency sampling method

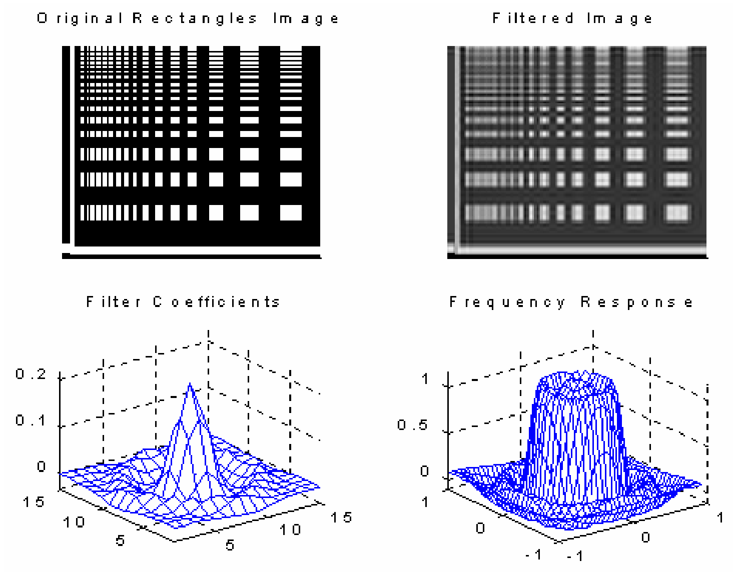


Fig. 5 The 2D FIR with 1D to 2D transformation method

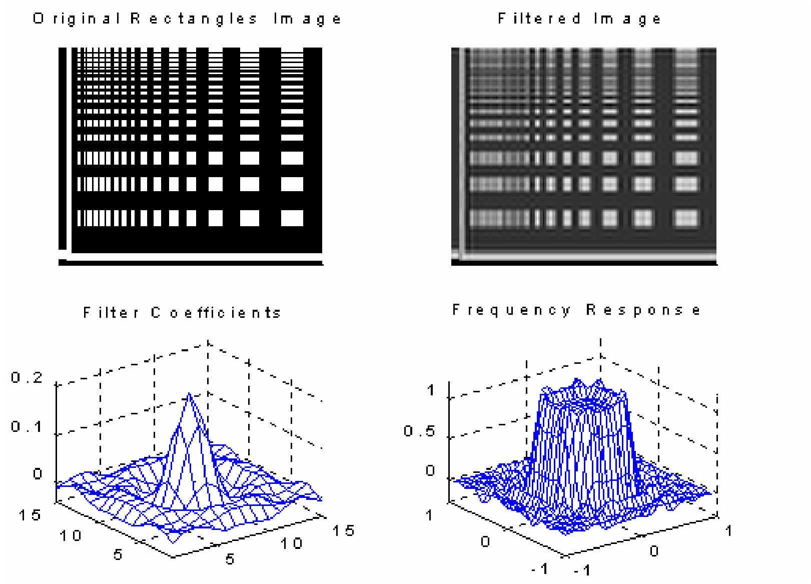


Fig. 6 The 2D FIR with FGA method

6 Discussion and Conclusions

Designing digital filters for medical applications requires highly linear phase response while keeping magnitude transfer function as close as possible to required characteristics. Symmetry of coefficients for the FIR filter guarantees linearity in phase. This is discovered by FGA without giving it any clue. The fitness (objective) equation only leads the FGA towards the characteristics of the optimal filter. The FGA could not get a better solution for size larger than 5 for the optimum number of coefficients (or possibly 6 in other fewer runs) in the 1D case and 15×15 coefficients in the 2D case. This is possibly because higher order filters tend to exhibit more non-linearity in their transfer function. This non-linearity may result in solutions far from the optimal rectangular transfer function, and hence, longer strings will have less fitness. There is an extent such as the order 4 or 5 where non-linearity is low enough. Of course, less non-linear solutions of order 2 or 3 are also bad solutions. The FGA simultaneously searches for the optimal number and values of coefficients in different pools in the search space. It is a highly parallel algorithm, but unfortunately, it is simulated with serial software and hardware architectures [21]. Table 1 shows a slight advantage of FGA over the other methods although the value of this error should be multiplied by 10^6 since the range of normalized frequency spreads over 512 points for both the magnitude and phase responses. After all, numbers in Table 1 are relative measures that are good for comparisons only. The table also shows the coefficients that were used in the design of all types of filters. Table 2 shows the optimal values for the number of coefficients for the FGA. The values shown in the table are the best out of 100 runs. It is shown that with five coefficients, the minimal error is reached.

In applications that are very sensitive to noise, the FGA can make a difference even with the slight advantage we have. Its strength comes from its ability to adapt and its flexibility to meet any design requirements depending on a user-made objective heuristic fitness function. Moreover, observing the filtering of the noisy input artificial ECG waves and the associated output waves, the FGA in general showed better performance than that of the standard window Kaiser technique. The FGA had better transfer functions for magnitude and kept linearity of phase. Future work includes testing with real-life ECG taken from MIT databases [22] and using a different ideal filter and different optimization technique such as Evolutionary Programming, Ant Colony Optimization and Swarm Intelligence [23–29]. Different forms of objective functions can be used. Some recent theories emphasizes the importance of the types of cost functions used in any optimization process and how success of any algorithm compared to the other one directly depends on cost function used [30].

References

[1] Davis, L.: Handbook of genetic algorithms. Van Nostrad, Reinhold (1991)

[2] Hassoun, M.H.: Fundamentals of Artificial Neural Networks. MIT Press, Cambridge (1995)

[3] Fraser, A.S.: Simulation of Genetic Systems by Automatic Digital Computers. I. Introduction, Australian J. Biological Sciences 10, 484–491 (1957)

[4] Bremermann, H.J.: Optimization through Evolution and Recombination, Self Organizing Systems 1962. In: Yovits, M.C., Jacobi, G.T., Goldstien, G.D. (eds.) Spartan Books, Washington DC, pp. 93–106 (1962)
[5] Holland, J.H.: Adaptation in Natural and Artificial Systems. University of Michigan Press, Ann Arbor (1975)
[6] Sornmo, L., Borjesson, P., Nygards, M., Pahlm, O.: A Method for Evaluation of QRS Shape Features Using a Mathematical Model for ECG. IEEE Trans. On Biomedical Engineering 28(10), 713–717 (1981)
[7] Goldberg, D.E.: Genetic Algorithms in Search Optimization and Machine Learning. Addison-Wesley, Reading (1989)
[8] Abu Zitar, R.A., Hassoun, M.H.: Neurocontrollers trained with rules extracted by a genetic assisted reinforcement learning system. IEEE Trans. on Neural Networks (4), 859–879 (1995)
[9] Nivon, V.: Two-Dimensional Zero Phase FIR Filter Design with non-uniform Frequency Sampling. Master's Thesis, Concordia University (1999)
[10] Cheney, E.W.: Introduction to Approximation Theory. McGraw-Hill, New York (1966)
[11] Lim, J.S.: Two-Dimensiona1 Signal and Image Processing. Prentice-Hall, Englewood Cliffs (1990)
[12] Dudgeon, D.E., Mersereau, R.M.: Multi-dimensional Digital Signal Processing. Prentice Hall, Englewood Cliffs (1984)
[13] Huang, T.S.: Two-Dimensional Windows. IEEE Trans. Audio Electracoustic AU-20(1), 88–89 (1972)
[14] McClellan, J.H.: The Design of Two-Dimensional Digital Filters by Transformations. In: Proc. 7th Annu. Princeton Conf. on Infom. Sci. and Syst., pp. 247–251 (1973)
[15] Bagchi, S., Mitra, S.K.: The Non-uniform Discrete Fourier Transform and its Applications in Filter Design. IEEE Trans. Circuits Syst. II 43, 422–444 (1996)
[16] Davis, L.: Adapting operator probabilities in genetic algorithms. In: Schaffer, J.D. (ed.) Proceedings of the Third International Conference on Genetic Algorithms, pp. 60–69. Morgan Kaufmann, San Mateo (1989)
[17] Nix, A.E., Vose, M.D.: Modeling Genetic Algorithms with Markov Chains. Annals of Mathematics and Artificial Intelligence, pp. 79–88 (1991)
[18] Lam, H.Y.F.: Analog and Digital Filters: Design and Realization. Prentice Hall, Englewood Cliffs (1979)
[19] Lu, W.-S., Antoniou, A.: Two-Dimensional Digital Filters. Marcel Dekker, New York (1992)
[20] Hu, J.V., Rabiner, L.R.: Design Techniques for Two-Dimensional Digital Filters. IEEE Trans. Audio Electroacoust AU-20(4), 249–256 (1972)
[21] Rumelhart, E.D., McClelland, J.L., The PDP Research Group: Parallel Distrib-uted Processing: Exploration in the Microstructure of Cognition, vol. 1. MIT Press, Cambridge (1986)
[22] http://db.csail.mit.edu
[23] Hinton, G.E., Nowlan, S.J.: How learning can guide evolution. Complex Sytems, 495–502 (1997)
[24] Fogel, D.B.: System Identification through Simulated Evolution: A Machine Learning Approach to Modeling. Ginn Press, Needham (1991)
[25] Abu Zitar, R.A.: Integer Coefficient FIR filter Design Using Evolutionary Programming Methods. In: Proceedings of the 4th Jordanian IEEE conference, pp. 93–97 (2001)

[26] Chin-Teng, L., Lee, C.S.G.: Neural fuzzy systems. Prentice Hall PTR, Upper Saddle River (1995)
[27] Cagnoni, S., Dobrzeniecki, A.B., Poli, R., Yanch, J.C.: Genetic-algorithm-based interactive segmentation of 3D medical images. Image and Vision Computing Journal 17(12), 881–896 (1999)
[28] Dorigo, M., Stutzle, T.: Ant Colony Optimization. MIT Press, Cambridge (2004)
[29] Kennedy, J., Eberhart, R.C., Shi, Y.: Swarm Intelligence. Morgan Kaufmann Publishers, San Francisco (2002)
[30] Wolpert, D., Macready, W.: No Free Lunch Theorems for Optimization. IEEE Trans. on Evolutionary Computation, pp. 67–82 (1997)

Author Index

Index