

Genetic and Evolutionary Computation for Image Processing and Analysis

Edited by: Stefano Cagnoni, Evelyne Lutton,
and Gustavo Olague



Genetic and Evolutionary Computation for Image Processing and Analysis

A print edition of this book can be purchased at
<http://www.hindawi.com/spc.8.html>
<http://www.amazon.com/dp/9774540018>

EURASIP Book Series on Signal Processing and Communications, Volume 8

Genetic and Evolutionary Computation for Image Processing and Analysis

Edited by: Stefano Cagnoni, Evelyne Lutton, and Gustavo Olaque

Hindawi Publishing Corporation
<http://www.hindawi.com>

A print edition of this book can be purchased at
<http://www.hindawi.com/spc.8.html>
<http://www.amazon.com/dp/9774540018>

EURASIP Book Series on Signal Processing and Communications

Editor-in-Chief: Alex Gershman

Editorial Board: Zhi Ding, Moncef Gabbouj, Peter Grant, Ferran Marqués, Marc Moonen, Hideaki Sakai, Giovanni Sicuranza, Bob Stewart, and Sergios Theodoridis

Hindawi Publishing Corporation

410 Park Avenue, 15th Floor, #287 pmb, New York, NY 10022, USA

Nasr City Free Zone, Cairo 11816, Egypt

Fax: +1-866-HINDAWI (USA Toll-Free)

© 2007 Hindawi Publishing Corporation

All rights reserved. No part of the material protected by this copyright notice may be reproduced or utilized in any form or by any means, electronic or mechanical, including photocopying, recording, or any information storage and retrieval system, without written permission from the publisher.

ISBN 978-977-454-001-1

A print edition of this book can be purchased at

<http://www.hindawi.com/spc.8.html>

<http://www.amazon.com/dp/9774540018>

Contents

I. Introduction

- | | |
|---|---|
| 1. Genetic and Evolutionary Computation for
Image Processing and Analysis,
<i>Stefano Cagnoni, Evelyne Lutton, and Gustavo Olaque</i> | 1 |
|---|---|

II. Low-level vision

- | | |
|---|-----|
| 2. Evolutionary multifractal signal/image denoising,
<i>Evelyne Lutton and Jacques Lévy Véhel</i> | 23 |
| 3. Sub-machine-code genetic programming for binary image analysis,
<i>Stefano Cagnoni, Monica Mordonini, and Giovanni Adorni</i> | 47 |
| 4. Halftone image generation using evolutionary computation,
<i>Kiyoshi Tanaka and Hernán Aguirre</i> | 65 |
| 5. Evolving image operators directly in hardware,
<i>Lukáš Sekanina and Tomáš Martínek</i> | 93 |
| 6. Variable-length compositional genetic algorithms for
the efficient implementation of morphological filters in
an embedded image processor,
<i>Ian P. W. Sillitoe and Andreas K. Magnusson</i> | 113 |
| 7. Autonomous model-based corner detection using
affine evolutionary algorithms,
<i>Gustavo Olaque and Benjamín Hernández</i> | 135 |

III. Midlevel vision

- | | |
|--|-----|
| 8. Evolution of an abstract image representation by
a population of feature detectors,
<i>Leonardo Bocchi</i> | 157 |
| 9. Genetic snakes: active contour models by genetic algorithms,
<i>Lucia Ballerini</i> | 177 |
| 10. Visual texture classification and segmentation by
genetic programming,
<i>Vic Ciesielski, Andy Song, and Brian Lam</i> | 195 |
| 11. A framework for the adaptation of image operators,
<i>Mario Köppen and Raul Vicente-Garcia</i> | 215 |

A print edition of this book can be purchased at

<http://www.hindawi.com/spc.8.html>

<http://www.amazon.com/dp/9774540018>

12. A practical review on the applicability of different evolutionary algorithms to 3D feature-based image registration,
Oscar Cordón, Sergio Damas, and José Santamaría 241
13. Image segmentation hybridizing variable neighborhood search and memetic algorithms,
Abraham Duarte, Ángel Sánchez, Felipe Fernández, and Antonio S. Montemayor 265
14. Model-based image analysis using evolutionary computation,
Jean Louchet 283

IV. High-level vision

15. Evolutionary feature synthesis for image databases,
Anlei Dong, Bir Bhanu, and Yingqiang Lin 309
16. Discovering of classification rules from hyperspectral images,
Arnaud Quirin and Jerzy Korczak 327
17. Genetic programming techniques for multiclass object recognition,
Mengjie Zhang 349
18. Classification by evolved digital hardware,
Jim Tørresen 371
19. Evolutionary photogrammetric network design,
Gustavo Olague and Enrique Dunn 393
20. Genetic algorithms and neural networks for object detection,
Mengjie Zhang 415
21. An evolutionary approach for designing multitarget tracking video systems,
Jesús García, Óscar Pérez, Antonio Berlanga, and José M. Molina 441

1

Genetic and Evolutionary Computation for Image Processing and Analysis

Stefano Cagnoni, Evelyne Lutton, and Gustavo Olaque

1.1. What is this book about?

After a long incubation in academia and in very specialized industrial environments, in the last ten to fifteen years research and development of image processing and computer vision applications have become mainstream industrial activities. Apart from the entertainment industry, where video games and special effects for movies are a billionaire business, in most production environments automated visual inspection tools have a relevant role in optimizing cost and quality of the production chain as well.

However, such pervasiveness of image processing and computer vision applications in the real world does not mean that solutions to all possible problems in those fields are available at all. Designing a computer application to whatever field implies solving a number of problems, mostly deriving from the variability which typically characterizes instances of the same real-world problem. Whenever the description of a problem is dimensionally large, having one or more of its attributes out of the “normality” range becomes almost inevitable. Real-world applications therefore usually have to deal with high-dimensional data, characterized by a high degree of uncertainty. In response to this, real-world applications need to be complex enough to be able to deal with large datasets, while also being robust enough to deal with data variability. This is particularly true for image processing and computer vision applications.

A rather wide range of well-established and well-explored image processing and computer vision tools is actually available, which provides effective solutions to rather specific problems in limited domains, such as industrial inspection in controlled environments. However, even for those problems, the design and tuning of image processing or computer vision systems is still a rather lengthy process, which goes through empirical trial-and-error stages, and whose effectiveness is mostly based on the skills and experience of the designer in the specific field of application. The situation is made even worse by the number of parameters which typically need to be tuned to optimize the performance of a vision system.

A print edition of this book can be purchased at

<http://www.hindawi.com/spc.8.html>

<http://www.amazon.com/dp/9774540018>

The techniques which are comprised under the term “soft computing” (namely, neural networks, genetic and evolutionary computation, fuzzy logic, and probabilistic networks) provide effective tools which deal specifically with the aforementioned problems. In this book, we focus on genetic and evolutionary computation (GEC) and try to offer a comprehensive view of how the techniques it encompasses can solve some of the problems which have to be tackled in designing image processing and computer vision applications to real-world problems.

In the rest of this chapter, we will offer a brief overview of the contents of the book. First, we will provide a quick introduction to the main EC paradigms, in order to allow subsequent chapters to concentrate more specifically on the description of each application and on the peculiarities of the approach they describe rather than on the basic approaches. Then, we will illustrate how the book, which does not necessarily require sequential reading, has been organized, to make it easier for readers to navigate through it and to find the topics which are more interesting to them.

1.2. When and how can genetic and evolutionary computation help?

From the point of view of artificial intelligence (AI), which focuses on mimicking the high-level “intelligent” processes which characterize living beings, genetic and evolutionary computation, as the other soft computing paradigms, is a way to provide computers with natural skills (self-improvement, learning, generalization, robustness to variability, adaptivity, etc.) based on nature-inspired paradigms. This point of view might seem too utopian to many, who might look upon natural processes, and even more on their imitation, as ill-defined and hardly deterministic process, which could be only partially kept under control by their users. However, things might look more convincing to a more down-to-earth audience, even if much less “romantic” and fascinating to others, if we turn to a more “mathematical” point of view stating that evolutionary computation comprises a wide set of powerful search and optimization methods, loosely inspired by the evolutionary processes which can be observed in nature. A third, intermediate, and very practical point of view, which is the one by which this book is addressing the topic, is an “engineering” point of view: GEC provides designers with a useful set of tools, inspired by natural evolution, which can help designing or refining the design of solutions to hard real-world problems.

Several factors are involved in the design of good solution to practical projects; the most important of which is definitely having extended “*a priori*” knowledge on the domain of interest. If one had full knowledge about the domain of interest, designing a solution would “just” require that the laws regulating its phenomenon be modeled in some manageable way. However, this is virtually never the case. As measurement theory teaches us, even the most indirect interaction with a phenomenon we are measuring is somehow able to alter the measure we are making. Therefore, having full knowledge of a problem domain means at least taking into account such perturbations. However, in general, the problem is by far more complicated. The representation we adopt is almost inevitably incomplete, as what we

actually observe derives from the overlap of several other concurrent events, most of which are unknown or unpredictable, with the phenomenon with which we are dealing. To make things worse, many problems do not allow for precise mathematical models to be defined, but they can be described only through extremely general concepts, whose instances are characterized by high variability.

In such situations (virtually always), there is no hope of finding a solution which will be equally good for all instances of the problem. Therefore, the actual skill of a designer is to find a good compromise which will be “good enough” in all or in most situations. This means being able to find the best solution, not only based on knowledge of the problem, but also relying on experimental clues which can be derived from observations, for the part of the problem for which knowledge is too limited or unavailable. These are typically skills that humans possess, at least as far as the domain of the problem is of limited extension, or subject to possible partial simplifications based (again) on knowledge.

When this is not the case, or when a nonexpert is facing such problems, the so-called *meta-heuristics* can provide effective help. Such a term refers to search methods which operate at a higher level of abstraction with respect to the problem under consideration, and can therefore be applied to a variety of tasks without requiring explicit knowledge (or requiring very limited knowledge) about the problem domain. Most often, these methods fit a general model to a dataset which describes the problem, by minimizing an error function or maximizing some score related to the quality of the solution they are searching, based on the performance of candidate solutions on a set of instances of the problem to be solved. This can be interpreted as “inductive learning,” if one feels more comfortable with the AI point of view, or as “function optimization,” if one prefers to use a more mathematical point of view. Among meta-heuristics, GEC techniques have attracted growing interest from several scientific communities. There are several reasons for that interest, which would require a ponderous book to be discussed extensively. In this section, we will just give a very general justification, which, however, is already by itself a good reason to approach such techniques.

In exploring a search space, that is, the domain of a function for which we are seeking some “interesting” points, such as the global maximum or minimum, there are two “extreme” strategies which can be adopted: blind/random search and greedy search. In the former, one explores the search space by randomly moving from one point to another relying just on luck. In the latter, one moves to the best point, which is accessible from the last visited one. In fact, resorting to some search method implies that we can only have knowledge about a limited portion of the search space at one time, which is typically a neighborhood of the last visited point. In random search, therefore, no sort of domain knowledge is exploited, and the space is just “explored,” while in greedy search, search is exclusively based on the exploitation of some, previously acquired or presently accessible (local) knowledge. For this reason, the problem of devising a good search strategy is often referred to as the “exploitation versus exploration dilemma.”

On the one hand, random search is the only way of exploring domains in which randomness is dominating and no assumptions can be made on the location

of good points based on local information. On the other hand, as soon as the search domain presents some regularities, exploiting local information can be crucial for success.

As will be shown in the next section, in which the main GEC paradigms will be described, each of these has both exploration (random) and exploitation (knowledge-based) components, associated to specific user-defined parameters which the user can set. This makes GEC paradigms particularly flexible, as they allow users to balance exploitation and exploration as needed.¹ This translates into highly effective and efficient searches by which good solutions can be found quickly.

The paradigms covered in the next sections are a nonexhaustive sample of GEC techniques, but wide enough to let the reader understand their basic principles and the algorithm variants which have been sometimes used by the authors of the following chapters.

1.3. A quick overview of genetic and evolutionary computation paradigms

The transposition into computers of the famous Darwin's theory consists of roughly imitating with programs the capability of a population of living organisms to adapt to its environment with selection/reproduction mechanisms. In the last forty years, various stochastic optimization methods have been based on this principle. *Artificial Darwinism* or *evolutionary algorithms* is a common name for these techniques, among which the reader may be more familiar with *genetic algorithms*, *evolution strategies*, or *genetic programming*.

The common components of these techniques are *populations* (that represent sample points of a search space) that evolve under the action of stochastic operators. Evolution is usually organized into *generations* and copies in a very simple way the natural genetics. The engine of this evolution is made of

- (i) *selection*, linked to a measurement of the quality of an individual with respect to the problem to be solved,
- (ii) *genetic operators*, usually *mutation* and *crossover* or *recombination*, that produce individuals of a new generation.

The efficiency of an evolutionary algorithm strongly depends on the parameter setting: successive populations (generations) have to converge toward what is wished, that is, most often the global optimum of a performance function. A large part of theoretical research on evolutionary algorithms is devoted to this delicate problem of convergence, as well as to trying to figure out what problem is easy or difficult for an evolutionary algorithm. Theoretical answers exist; these algorithms converge [2, 12, 26, 40, 55]; but other important practical questions, like convergence speed, remain open. One can therefore say that the interest into evolutionary techniques is reasonably funded theoretically, which justifies forty years of successful experimental developments.

¹On the actual meaning of "as needed" in the case of genetic and evolutionary search, much can be debated, but let us keep our discussion as general as possible.

Moreover, evolutionary techniques are zero-order stochastic optimization methods, that is, no continuity nor derivability properties are needed: the only information which is required is the value of the function to be optimized at the sample points (sometimes, even an approximation can be used). These methods are thus particularly adapted to very irregular, complex, or badly conditioned functions. Their computation time, however, can be long.

Evolutionary techniques are usually recommended when other more classical and rapid methods fail (for very large search spaces, mixed variables, when there are many local optima, or when functions are too irregular). Other problems, like dynamic or interactive problems, can also be addressed with evolutionary algorithms; and finally, these methods can be successfully hybridized with classical optimization methods (e.g., gradient descent, tabu search).

Despite the attractive simplicity of an evolutionary process, building an efficient evolutionary algorithm is a difficult task, as an evolutionary stochastic process is very sensitive to parameter and algorithm setting. The elaboration of an efficient evolutionary algorithm is based on a good knowledge of the problem to be solved, as well as on a good understanding of the evolution mechanisms. A “black box” approach is definitely not recommended.

Industrial “success-stories” are numerous and various,² also in the domain of image analysis and robot vision.

1.4. Basic concepts of artificial evolution

Evolutionary algorithms have borrowed (and considerably simplified!) some principles of natural genetics. We thus talk about *individuals* that represent solutions or points of a search space, also called *environment*. On this environment, a maximum of a *fitness function* or *evaluation function* is then searched.

Individuals are usually represented as codes (real, binary, of fixed or variable size, simple or complex), they are *chromosomes* or *genomes*, that is, *genotypes*. The corresponding solutions (i.e., the vectors of the search space) are *phenotypes*. An evolutionary algorithm evolves its population in a way that makes individuals more and more *adapted* to the environment. In other terms, the fitness function is *maximized*.

What is described below is a basic canvas; a “canonic” evolutionary algorithm. Real-life applications are of course much more complex, with the main problem being to adapt, or even create, operators that correspond to the problem at hand.

1.4.1. The evolution loop

The first element is a generation loop of populations of individuals, with each individual corresponding to a potential solution to the considered problem (see Figure 1.1 and [17, 5, 11, 16, 38]).

²See [17, pages 126–129] for examples of applications developed before 1989, and on <http://evonet.lri.fr> or [1, 10, 21, 29, 43, 54] for more recent applications.

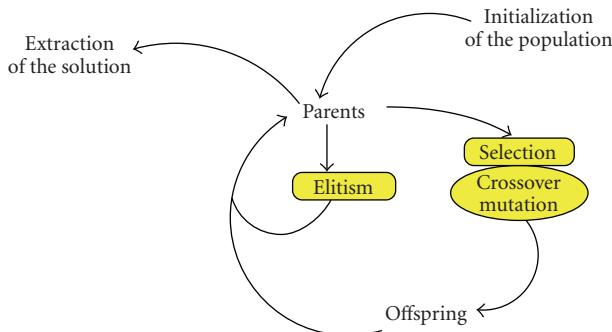


FIGURE 1.1. Organigram of a simple evolutionary algorithm.

Initialization is usually random (other strategies are sometimes used, particularly in complex or high-dimensional search spaces). Initial solutions (obtained, e.g., using a classical optimization technique) can also be integrated into the initial population. If the initial population content has theoretically no importance (the limit distribution of such a stochastic process is always the same), it is noticed experimentally that initialization has a big influence on variance of the results and speed of convergence. It is often very efficient to inject “*a priori*” information about the problem at the initialization stage.

Selection decides which individuals of the current population are going to reproduce. It is based on the notion of “quality” of an individual, embedded in the *fitness function*.

The main parameter of selection is the *selective pressure*, usually defined as the quotient of the probability of selecting the best individual over the probability of selecting an average individual. The selective pressure has a strong influence on the *genetic diversity* of the population, and consequently on the efficiency of the whole algorithm. For instance, an excessive selection pressure may produce a rapid concentration of the population in the vicinity of its best individuals, with a risk of premature convergence toward a local optimum.

The simplest selection is the *proportional selection*, implemented with a biased random shot, where the probability of selecting an individual is directly proportional to its fitness value:

$$P(i) = \frac{\text{fitness}(i)}{\left(\sum_{k=1}^{\text{PopSize}} \text{fitness}(k) \right)}. \quad (1.1)$$

This scheme does not allow to control the selective pressure. Other—and more efficient—selection schemes are, for example,

- (i) *scaling*, that linearly scales the fitness function at each generation in order to get a maximal fitness that is C times the average fitness of the current population. C measures the selective pressure, usually fixed between 1.2 and 2 [17];

- (ii) *ranking*, that allocates to each individual a probability that is proportional to its rank in a sorted list according to fitness;
- (iii) *tournament*, that randomly selects T individuals in the population (independently to their fitness values) and chooses the best. The selective pressure is linked to the size T of the tournament.

Reproduction generates offspring. In the canonic scheme “à la Goldberg” [17], 2 parents produce 2 children; a number of parents equal to the desired number of offspring is thus selected. Of course, many other less-conventional schemes can be programmed (2 parents for 1 child, n parents for p children, etc.).

The two main *variation operators* are crossover, or recombination, that recombines genes of parents, and mutation, that slightly perturbs the genome. These operations are randomly applied, based on two parameters: crossover probability p_c and mutation probability p_m .

Intuitively, selection and crossover tend to concentrate the population near “good” individuals (information exploitation). On the contrary, mutation limits the attraction of the best individuals in order to let the population explore other areas of the search space.

Evaluation computes (or estimates) the quality of new individuals. This operator is the only one that uses the function to be optimized. No hypothesis is made on this function, except for the fact that it must be used to define a probability or at least a rank for each solution.

Replacement controls the composition of generation $n + 1$. Elitism is often recommended for optimization tasks in order to keep the best individuals from a population into the next one. Usual strategies directly transmit a given percentage of the best individual in the next population (e.g., generation gap of [27]). Evolution strategies (μ, λ) and $(\mu + \lambda)$ [4, 22, 23] produce λ offspring of a population of μ individuals. The “,” strategy controls elitism via the difference $\mu - \lambda$ (the $\mu - \lambda$ best individuals are kept and completed by λ offspring), while the “+” strategy is more adaptive: from an intermediate population of size $\mu + \lambda$, made of the current population of size μ and λ offspring, the μ best individuals are selected for the next generation.

In the case of parallel implementations, it is sometimes useful to use another scheme instead of the one based on generations: the *steady state* scheme adds directly each new individual in the current population via a replacement operator (reverse selection) that replaces bad individuals of the current population by new ones.

Stopping the evolution process at the right moment is crucial from a practical viewpoint; but if little or no information is available about the value of the searched optimum, it is difficult to know when to stop. A usual strategy is to stop evolution after a fixed number of generations, or when stagnation occurs. It is also possible to test the dispersion of the population. A good control of the stopping criterion obviously influences the efficiency of the algorithm, and is as important as a good setting of evolution parameters (population size, crossover and mutation probabilities, selective pressure, replacement percentage, etc.).

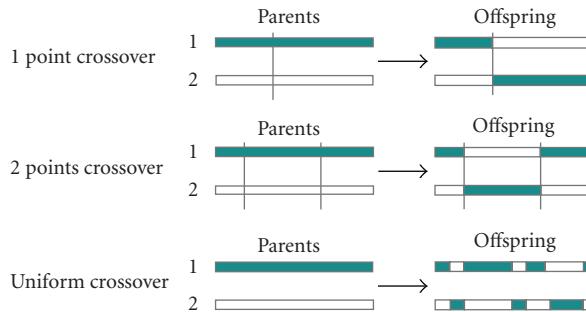


FIGURE 1.2. Binary crossover.

An evolutionary algorithm (EA) is a partially blind search algorithm, whose blind/random component has to be cleverly tuned, as a function of what is known as “*a priori*” about the problem to be solved: too much randomness is time consuming, and too little may let the process be blocked in a local optimum.

1.4.2. Representations and operators

Genetic operators directly depend on the choice of the representation, which, for example, makes the difference between genetic algorithms, evolution strategies, genetic programming, and grammatical evolution. We quickly present below the most usual representations, operators, selection and replacement schemes. Many other schemes for nonstandard search spaces can be found in the literature as for instance, list or graph spaces.

1.4.2.1. Discrete representation

Genetic Algorithms are based on the use of a binary representation of solutions, extended later to discrete representations.³

Each individual of the population is represented by a fixed-size string, with the characters (genes) being chosen from a finite alphabet. This representation is obviously suitable for discrete combinatorial problems, but continuous problems can be addressed this way thanks to a sampling of the search space. In this case, the sampling precision (related to the chromosome length) is an important parameter of the method [34].

The most classical crossover operators used in optimization tasks are described in Figure 1.2. The *one-point crossover* randomly chooses a position on the chromosome and then exchanges chain parts around this point. The *two-point crossover* also exchanges portions of chromosomes, but selects two points for the exchange. Finally, the *uniform crossover* is a multipoint generalization of the previous one: each gene of an offspring is randomly chosen between the parents’ genes

³Even if there exists now real encoded genetic algorithms, the discrete encoding is the historical characteristic of the “genetic algorithms trend.”

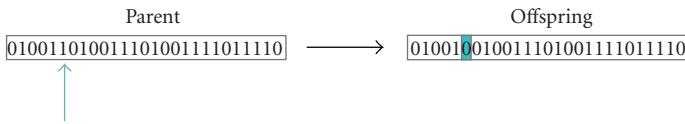


FIGURE 1.3. Binary mutation.

at the same position. Other specialized crossovers exist, like in the case of travelling salesman problems or scheduling problems, which take into account the specific structure of the gene encoding.

The classical binary mutation flips each bit of the chromosome with a probability p_m (see Figure 1.3). The mutation probability p_m is usually very low and constant along the evolution, but some schemes exist where the mutation probability decreases along generations.⁴

1.4.2.2. Continuous representation

The continuous representation, or real representation, is historically related to evolution strategies. This approach performs a search in \mathbb{R}^n or in a part of it. The associated genetic operators are either extensions to continuous space of discrete operators, or directly continuous operators.

The *discrete crossover* is a mixing of real genes of a chromosome, without change of their content. The previous binary crossover operators (one point, two points, uniform) can thus be adapted in a straightforward manner.

The benefit of continuous representation is surely better exploited with specialized operators, that is, *continuous crossover* that mixes more intimately the components of the parents vectors to produce new individuals. The *barycentric* crossover, also called *arithmetic*, produces an offspring x' from a couple (x, y) of \mathbb{R}^n thanks to a uniform random shot of a constant α in $[0, 1]$ (or $[-\epsilon, 1 + \epsilon]$ for the BLX- ϵ crossover) such that

$$\forall i \in 1, \dots, n, \quad x'_i = \alpha x_i + (1 - \alpha) y_i. \quad (1.2)$$

The constant α can be chosen once for all coordinates of x' , or independently for each coordinate.

The generalization to a crossover of more than 2 parents, or even the entire population set (“global” crossover) is straightforward [45].

Many mutation operators have been proposed for the real representation. The most classical is the *Gaussian mutation*, that adds a Gaussian noise to the components of the individual. It requires that an additional parameter, σ , the standard deviation of the noise, be tuned:

$$\forall i \in 1, \dots, n, \quad x'_i = x_i + N(0, \sigma). \quad (1.3)$$

⁴It has been theoretically proved that a mutation-only genetic algorithm converges towards the global optimum of the search space only if p_m decreases according to a logarithmic rate [12].

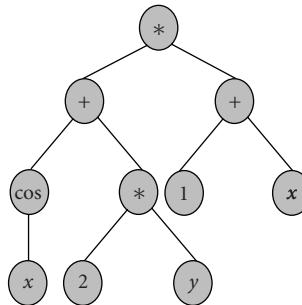


FIGURE 1.4. Example of a tree representation of the function $((\cos(x) + 2 * y) * (1 + x))$.

Tuning σ is relatively complex (too small, it slows down evolution; too large, it affects negatively the convergence of the algorithm). Various strategies that make σ vary along evolution have been tested: σ as a function of time or fitness value, as a function of the direction of search (anisotropic mutations), or even self-adaptive (i.e., with σ being considered an additional parameter, i.e., evolved by the algorithm). Other studies have been performed on the use of non-Gaussian noise.

1.4.2.3. Trees representations

Genetic programming (GP) corresponds to a representation of variable-length structures as trees. GP has been initially designed to handle LISP programs [29], in order to create programs able to solve problems for which they were not explicitly programmed. The richness and versatility of the variable-size tree representation (see Figure 1.4) are at the origin of the success of GP. Many optimization, command or control problems can be formulated as a program induction problem. Recently in the computer vision domain, genetic programming has been shown to achieve human competitive results [53].

A GP algorithm explores a search space of recursive programs made of elements of a function set, of a variable set, and of a terminal set (data, constants).⁵ Individuals of the population are programs that, when executed, produce the solution to the problem at hand.

Crossovers are often subtree exchanges. Mutations are more complex, and several mutations have to be used, producing different types of perturbations on the genome structure: suppression/addition of a node, modification of the content of a node, mutation of the constants (continuous values), and mutation of discrete variables.

Applications of genetic programming are numerous, for example, in optimal control, in trajectory and action planning in robotics, or in symbolic regression (search for a mathematical formula that approximates a finite set of samples).

⁵A current problem of GP is the so-called *bloat*, that is, the saturation of the memory space due to a disproportionate growth of the trees sizes along evolution. A good way to avoid this effect is to limit genome sizes [32, 48].

1.5. Doing more than optimization

Evolving a population on a search space according to the previous principles allows not only to localize the global optimum of a complex function (theoretical proofs exist, see [2, 12, 26, 40, 55]), but also to gain more information on the function and its search space.

For instance, if the function to be optimized is multimodal, slight modifications of the evolution loop allow to make the population converge into subpopulations localized on “niches” corresponding to each optimum. These methods control the diversity of the population, or implement a resource-sharing mechanism between neighbor individuals [17, 18] to favor the emergence of distinct species. The definition of an interchromosomes distance is then necessary.

It is also possible to consider a problem as a collective learning task, with the searched solution being built from the whole set of individuals of an evolved population, and not only from its single best individual. The most famous techniques of this type are classifier systems [7], the Parisian approach [9, 41], cooperative coevolution [42], and techniques based on social insect colonies, like ant colony algorithms (ACO) [13, 14].

The Parisian approach has, for example, produced applications in text retrieval [30, 31], in art and design [8, 15], or even real-time applications (stereo vision using the “fly algorithm” [36]), which is noticeable for algorithms that have the reputation of being big CPU time consumers!

Moreover, in some applications, the precise identification of quantities to be optimized is sometimes difficult, especially in cases where there exist several judgment criteria, possibly contradictory (e.g., maximize the resistance of a mechanical part, while minimizing its weight and its cost). These optimizations are even more complex to handle if there is no way of estimating the relative importance of each criterion. One thus consider multicriterion optimization, without giving any priority to the various criteria. The solution to a multicriterion problem is thus a set, the *Pareto front*, of optimal compromises. The idea of using evolutionary techniques to find the Pareto front of a multicriterion problem is quite natural, and based on a small modification of the classical evolutionary scheme. More precisely, the selection operator is adapted in order to push the population toward the Pareto front, while maintaining diversity to provide a good sampling of the front. Once again, diversity control is a key point. A comparative study of evolutionary methods for multicriteria optimization can be found in [56].

Finally, if what we wish to optimize is not measurable with a mathematical function or a computer procedure (e.g., the simple notion of “being satisfied”), one has to put a human in the evolutionary loop, that is, consider *interactive evolutionary algorithms*. The first studies in this domain [3, 46, 47, 51] were oriented toward artistic design (e.g., numerical images or 3D shapes synthesis). Much work concerns now various application domains, where quantities to be optimized are linked to subjective rating (visual or auditory). Characteristic work are, for instance, [50] for adapting hearing aids, [28] for the control of robot arm to provide smooth and human-like movements, or [39] for the design of HTML pages. A review of this broad topic can be found in [49] or in [44].

1.6. Contents

In this section, we briefly introduce the contents of the book, according to the logical subdivision of the volume into three main sections, which are dedicated to low-level, midlevel, and high-level visions, respectively.

1.6.1. Low-level vision

Early stages of image processing—low-level vision tasks—have been largely investigated for many years. Typical tasks are image filtering, smoothing, enhancement or denoising, lightness computation, edge and singular point detection, resampling, quantization, and compression. Low-level processing usually takes into account close neighborhood relations in images, morphologic properties, or even 3D geometry (including problems of camera distortion and partial occlusion).

This topic remains, however, a source of challenging problems, as the quality of outputs is crucial for the whole computer vision chain. Sophisticated mathematical theories and statistical methods have been developed in recent years, that are a source of complex optimization problems. Additionally, new constraints for embedded, real-time computer vision systems necessitate robust and flexible as well as cost-effective algorithms.

In this section of the book, various examples show the benefit of using artificial evolution techniques to tackle complex low-level tasks, impossible to address with classical optimization techniques, improving versatility, precision, and robustness of results. We will see also in the sequel that real-time or quasireal-time processing can be attained with evolutionary techniques, in spite of the computation time-gluttony reputation of these techniques.

The first chapter, entitled “Evolutionary multifractal signal/image denoising” by Lutton and Levy Vehel, deals with enhancement or denoising of complex signals and images, based on the analysis of local Hölder regularity (multifractal denoising). This method is adapted to irregular signals that are very difficult to handle with classical filtering techniques. Once again, the problem of denoising has been turned into an optimization one: searching for a signal with a prescribed regularity that is as near as possible to the original (noisy) one. Two strategies are considered: using evolution as a pure stochastic optimizer, or using interactive evolution for a metaoptimization task. Both strategies are complementary as they allow to address different aspects of signal/image denoising.

The second chapter, entitled “Submachine-code genetic programming for binary image analysis” by Cagnoni, Mordonini, and Adorni, addresses issues related to quasireal-time image processing. The authors present a solution that exploits in a clever way the intrinsic parallelism of bitwise instructions of sequential CPUs in traditional computer architectures. In other words, genetic programming is used to optimize a set of binary functions, that are used as binary classifiers (submachine-code genetic programming, SmcGP). The application considered is license-plate recognition, which is composed of two tasks: license-plate localization in the image (region-based segmentation), and low-resolution character recognition. Both are formulated as classification tasks. GP-based techniques are

compared to neural net techniques for the same tasks. SmcGP classifiers are almost as precise as the LVQ neural net used as reference classifier, but with processing times that are about 10 times faster. Using SmcGP in the preprocessing stage of a license-plate recognition system has also been proved to improve robustness. Additionally, the functions evolved with SmcGP can be easily integrated in embedded systems, as Boolean classification functions as those evolved by SmcGP can be directly implemented in digital hardware.

The third chapter, entitled “Halftone image generation using evolutionary computation” by Tanaka and Aguirre, investigates the problem of generating halftone images. Using a genetic algorithm has been proven to be beneficial. However, as this technique is computationally expensive, it is necessary to build improved GA schemes for practical implementations. Compromises have to be found in order to be able to use GA-based techniques in practical implementations. This chapter is a good example of an adaptation of the genetic operators and evolution scheme to specificities of the genome (image blocks specialized operators, fine design of the functions to be optimized, and multiobjective approach).

The fourth chapter, entitled “Evolving image operators directly in hardware” by Sekanina and Martinek, considers the problem of automatic designing of image filters based on an evolvable hardware system (FPGA). The idea is to be able to automatically design filters when corrupted, and original images are supplied by the user. The learning problem is turned into an optimization problem, that is to find the filter that minimizes the difference between the corrupted and original images of the training set. The filters are combined from elementary components (minimum, maximum, average, and other logic functions over two pixels) using Cartesian genetic programming. Examples are provided for noise removal and edge detection tasks. The originality of this work is that everything is implemented on hardware, that is, the filters as well as the evolutionary algorithm itself. The advantage of such an implementation is the performance (a filter can be evolved in 20 seconds on an FPGA operating a 100 MHz!), and for some applications it is thus possible to approach real-time evolutionary design. A precise analysis of the influence of parameters setting on quality and generality of filters and on the time of evolution is also presented.

The fifth chapter, entitled “Variable-length compositional genetic algorithms for the efficient implementation of morphological filters in an embedded image processor” by Sillitoe and Magnusson, is also related to high-speed binary image processing and embedded vision systems. This chapter describes the implementation of morphological image filters using a variable-length steady-state GA on a high-speed image processor. A specific mechanism to maintain diversity has been developed to cope with the rugged fitness landscape induced by the processor architecture. The aim of the optimization procedure is to map the original filter specification into a reduced sequence of machine-specific operators and connectives. This chapter addresses an interesting point about variable-length genomes: the so-called “compositional operator” is applied only when a stagnation is detected, which has a consequence that evolution of genome content has the priority over structure evolution. In the fitness evaluation, there is also an additional term

that promotes individuals which implement elements of the solution not commonly found in the current population.

The sixth chapter, entitled “Autonomous model-based corner detection using affine evolutionary algorithms” by Olague and Hernández, is an example of an approach based on a versatile nonlinear corner model whose parameters are estimated via an optimization procedure (resolution of an inverse problem) based on an EA. Additionally, new genetic operators based on homogeneous matrix representations have been designed according to the specific corner model. Comparisons have been done with other optimization methods proving that EA provides a more robust estimation technique. This is an example of the capability of EA to handle nonlinear models, which allow to cope with more complex photogrammetric models.

1.6.2. Midlevel vision

Midlevel vision algorithms, as the name suggests, provide the necessary connection between low-level algorithms and high-level ones. The former are aimed at emulating the innate specificity of human perception, which includes processing tasks occurring mostly at an unconscious level, while the latter, which can be related to cognitive tasks rather than to perceptual ones, implement the conscious interpretation of the scene under observation, based not only on information extracted by perceptual elements, but mainly on knowledge-based processes based on the observer’s own experience.

The aim of midlevel tasks is, therefore, to translate perceptual representation of the image into a symbolic representation on which high-level reasoning processes can operate to achieve full understanding of the contents of the scene represented within the image.

Image segmentation is definitely the most relevant and recurring task within midlevel algorithms, and can almost be identified with the whole class, if its definition as “grouping of perceptual information according to some uniformity/ classification criterion” is given the slightly more flexible interpretation as “integration of basic image elements into “more meaningful” and complex structures to which a symbolic meaning can be attached.” Another popular application of midlevel vision is image registration.

The chapters in this section describe several ways in which the design of midlevel vision algorithms can be supported by different EC techniques, in different domains of application. They show how problems can be tackled by computer vision-centered approaches, in which EC techniques are used essentially as optimization tools, as well as by EC-centered approaches in which problems are observed from a substantially different point of view, directly induced by the features of the EC technique which is adopted.

In the chapter entitled “Evolution of an abstract image representation by a population of feature detectors,” Bocchi presents an artificial life-inspired approach based on an evolutionary network of entities which identify and track “key” points in the image. Each entity “learns” to localize one of the features which

are present in the image, and coordinates with neighboring entities to describe the spatial relationships among the features. The population implements both a short-term migration of the units to dislocate on an unknown image, and a long-term adaptation to improve the fitness of the population to the environment which is present on all images in the data set. Once adaptation is complete, the feature vectors associated to each entity represent the features which have been identified in the image set, and the topological relations among those features in the image are mirrored in the neighborhood relations among the corresponding individuals. A sample toy problem is used to show the basic properties of the population, where the population learns to reproduce a hand-written letter, while an application to the biomedical domain (identification of bones in a hand radiogram) measures the performances of the architecture in a real-world problem.

The chapter by Ballerini, entitled “Genetic snakes: active contour models by genetic algorithms,” reviews and extends the definition of “genetic snakes,” active contour models optimized with a procedure based on genetic algorithms. Originally developed for application to problems in computer vision and computer graphics, snakes have been extensively applied in medical image analysis in problems including segmentation, shape representation, matching, and motion tracking, and have achieved considerable popularity. However, the application of snakes to extract region of interest suffers from some limitations. In fact, a snake is an energy-minimizing spline, and the classical model employs variational calculus to iteratively minimize the energy. There may be a number of problems associated with this approach such as algorithm initialization, existence of local minima, and selection of model parameters. “Genetic snakes” have been shown to be able to overcome some limits of the classical snakes and have been successfully applied to segment different kinds of images. In the chapter under consideration, new problem-specific energy functionals are used in the fitness function driving the evolution of snakes. Experimental results on synthetic images as well as on real images are conducted with encouraging results.

Ciesielski, Song, and Lam, in the chapter entitled “Visual texture classification and segmentation by genetic programming,” show that genetic programming can be used for texture classification in three ways: (a) as a classification technique for feature vectors generated by conventional feature extraction algorithms, (b) as a one-step method that bypasses feature extraction and generates classifiers directly from image pixels, and (c) as a method of generating novel feature extraction programs. All of the above approaches have been tested on a number of difficult problems drawn from the Brodatz texture library. Authors show, in particular, how the one-step classifiers can be used for fast, accurate texture segmentation. In doing so, they show that the use of the genetic programming techniques can overcome some of the drawbacks, which are briefly listed here, affecting the application of traditional texture analysis techniques. Firstly, it is impossible to define a universal set of optimal texture features, which causes the need for a trial- and error-process for each new texture classification/segmentation task, to find a feature set that works well. Secondly, some of the approaches generate an enormous number of features which calls for effective techniques for dimensionality reduction in feature space.

Thirdly, most of the texture feature extraction algorithms are computationally expensive and require the generation of Fourier-type transforms or other complex intermediate data structures and then additional computation on these structures.

Another evolutionary approach to texture classification is presented by Koeppen and Garcia, in the chapter entitled “A framework for the adaptation of image operators.” The chapter describes a framework, which allows for the design of texture filters for fault detection. The framework is based on the 2D-lookup algorithm, where two filter output images and a 2D-lookup matrix are used as inputs. The algorithm runs through all pixel positions in both images, and takes the gray value pair at the corresponding position as coordinates in the matrix. The value stored in this matrix position is used as the return value in the result image at the actual position. Having n operators available, there are $n^*(n - 1)/2$ possible ways to select a pair of operators, and this number grows even more if the operation allows for internal parameter settings. An evolutionary procedure is used to select the best operation pair. The chapter also introduces a generic design method which builds more complex operators from simple ones, which is based on genetic programming, the best established procedure so far to allow for such an optimization as well. The framework can be extended in various ways; two of which are also presented in the chapter.

In the chapter entitled “A practical review on the applicability of different evolutionary algorithms to 3D feature-based image registration,” Cordón, Damas, and Santamaría introduce image registration (IR), the process of finding the optimal spatial transformation (e.g., rigid, similarity, affine, etc.) achieving the best fitting/overlaying between two (or more) different images related by the latter transformation, measured by a similarity metric function. IR is presently a very active research area in the computer vision community. The chapter discusses the basic problem and its components, and addresses the recent interest on applying evolutionary algorithms to image registration, considering different approaches to the problem and describing the most relevant applications. A practical review focusing on feature-based IR considering both evolutionary and nonevolutionary approaches is also developed. The review is supported by a broad experimentation of those IR methods on the registration of some magnetic resonance images of human brains. To the best of our knowledge, this is the first review which compares different evolutionary and nonevolutionary techniques reporting results obtained on the same test images.

The chapter by Duarte, Sánchez, Fernández, and Montemayor, entitled “Image segmentation hybridizing variable neighborhood search and memetic algorithms,” introduces a new hybrid evolutionary algorithm as a graph-based image segmentation technique to improve quality results. The method proposed in this chapter can be considered as region-based, resulting in a k -region decomposition of the scene. The underlying model and approach to solving image segmentation as a graph-partitioning problem is related to Shi and Malik’s work. They use a computational technique based on a generalized eigenvalue problem for computing the segmentation regions. This algorithm combines oversegmented regions using a low-level hybridization between a variable neighborhood search

and a memetic algorithm. An oversegmented version of an original image is represented as an undirected weighted graph. In this graph, nodes are the image regions and the arcs together with their associated weights are defined using local information. The graph construction is modeled as an alternative region adjacency graph; here called modified region adjacency graph.

Finally, Jean Louchet, in the chapter entitled “Model-based image analysis using evolutionary computation,” shows how evolution strategies can actually widen the scope of Hough transform generalizations and how some of their variants and extensions, in particular the Parisian approach, can efficiently solve real-time computer vision, sensor fusion, and robotics problems with little reference to more traditional methods. In the first part of this chapter, the author shows, through several example problems, that evolution strategies give a new life to model-based image analysis, thanks to their ability to efficiently explore complex model parameter spaces. In the second part, the Parisian variant of evolution strategies is considered, showing, through an application to stereo vision (the “fly algorithm”), that it provides fast and efficient algorithms with interesting real-time and asynchronous properties, specially valuable in autonomous robotics applications and image analysis in changing environments.

1.6.3. High-level vision

High-level vision is devoted to the study of how the cognitive approach is implemented in the computer. Several tasks are related to cognitive or mental tasks such as content-based image retrieval, recognition, identification, 3D scene analysis, and design.

This last stage of the computer vision chain is as the two previous ones a rich source of challenging problems, in which evolutionary algorithms achieve successful applications with innovative solutions.

In this section of the book, seven chapters have been included to illustrate how evolutionary algorithms could be applied to solve complex high-level vision tasks. The applications are centered on recognition, detection, design of photogrammetric networks, and classification tasks. These chapters show a general balance between the use of computer vision and evolutionary computation knowledge.

The first chapter, entitled “Evolutionary feature synthesis for image databases,” by Dong et al., describes a genetic programming approach used in synthesizing feature vectors in order to improve the performance of content-based image retrieval. The advantage of dimensionality reduction, as well as the fact that the genetic programming approach does not assume any class distribution in the original feature space, gives distinct advantage over the linear transformation and the support vector machine approaches. Results over several image datasets have demonstrated the effectiveness of genetic programming in improving image retrieval performance.

The second chapter, by Quirin and Korczac, entitled “Discovering of classification rules from hyperspectral images,” presents a learning classifier system

applied to remote sensing images in order to find the best set of rules without human intervention. The proposed system has been validated with a comparison to other approaches such as neural networks and supports vector machines. Finally, the results have shown the potential of applying learning classifier systems to the discovery of rules in remote sensing images.

The third chapter, entitled “Genetic programming techniques for multiclass object recognition,” by Zhang, proposes the use of dynamic class boundary detection methods to improve the static method that was previously applied in the domain of multiclass object detection using genetic programming. The results confirm that a dynamic approach could classify better the objects if the classes are arranged in an arbitrary order or when the classification problems become more difficult.

The fourth chapter, entitled “Classification by evolved digital hardware,” by Tørresen, presents an evolvable hardware approach based on a divide-and-conquer strategy called incremental evolution, which aims to improve the solution by dividing the problem domain while incrementally evolving the hardware system. This is also called “increased-complexity evolution.” Thus, the evolution is undertaken individually on a set of small systems in order to spend less effort than for evolving a single big system. Examples are provided to show how to evolve both a prosthetic hand controller circuit and for classifying numbers on speed limit signs. The results illustrate that this is a promising approach for evolving systems in the case of complex real-world problems.

The fifth chapter, by Olague and Dunn, entitled “Evolutionary photogrammetric network design,” addresses the problem of configuring an optimal photogrammetric network in order to measure a complex object with high accuracy. The fitness function is implemented through an analytical uncertainty analysis, as well as the classical bundle adjustment. The optical and environmental constraints are incorporated in the evolutionary process. The strategy proposed here has shown how human-competitive designs could be achieved in the case of a large number of cameras, considering multiple competing constraints until the best acceptable configuration is found. A number of experiments are provided to illustrate the applicability of the simulator.

The sixth chapter, by Zhang, entitled “Genetic algorithms and neural networks for object detection,” describes a domain-independent approach to multiple class object detection based on training a neural network classifier on cutouts of the objects of interest and then refining the network weights using a genetic algorithm. The results show promising results for the case of retinal pathologies in which the proposed technique is competitive with statistical and neural networks approaches.

Finally, the seventh chapter, entitled “An evolutionary approach for designing multitarget tracking video systems,” proposes the use of evolution strategies for the development of an aircraft surveillance system. The proposed methodology apply the concept of partial evaluation using aggregation operators to build the evaluation function. This analogy is the base for stating the problem in terms of optimization in order to give an appropriate output of the video surveillance

system under different situations. Several experiments are provided to illustrate the applicability of the proposed technique with respect to real situations.

Bibliography

- [1] J. Albert, F. Ferri, J. Domingo, and M. Vincens, "An approach to natural scene segmentation by means of genetic algorithms with fuzzy data," in *Proceedings of the 4th National Symposium in Pattern Recognition and Image Analysis*, P. de la Blanca, Ed., pp. 97–113, Singapore, September 1992.
- [2] L. Altenberg, "Evolutionary computation models from population genetics. part 2: a historical toolbox," in *Proceedings of the IEEE Congress on Evolutionary Computation (CEC '00)*, San Diego, Calif, USA, July 2000.
- [3] P. J. Angeline, "Evolving fractal movies," in *Proceedings of the 1st Annual Conference on Genetic Programming*, J. R. Koza, D. E. Goldberg, D. B. Fogel, and R. L. Riolo, Eds., pp. 503–511, MIT Press, Cambridge, Mass, USA, 1996.
- [4] T. Bäck and H. P. Schwefel, "An overview of evolutionary algorithms for parameter optimization," Tech. Rep., University of Dortmund, Dortmund, Germany, 1992.
- [5] W. Banzhaf, "Interactive evolution," in *Handbook of Evolutionary Computation*, Oxford University Press, Oxford, UK, 1997.
- [6] A. Boumaza and J. Louchet, "Dynamic flies: using real-time parisian evolution in robotics," in *Applications of Evolutionary Computing, EvoWorkshops: EvoCOP, EvoFlight, EvoIASP, EvoLearn, and EvoSTIM*, E. J. W. Boers, J. Gottlieb, P. L. Lanzi, et al., Eds., vol. 2037 of *Lecture Notes in Computer Science*, pp. 288–297, Springer, Como, Italy, April 2001.
- [7] L. Bull and T. C. Fogarty, "Co-evolving communicating classifier systems for tracking," in *Artificial Neural Networks and Genetic Algorithms*, pp. 522–527, Springer, Wien, Austria, 1993.
- [8] J. Chapuis and E. Lutton, "ArtiE-fract: interactive evolution of fractals," in *Proceedings of the 4th International Conference on Generative Art (GA '01)*, Milano, Italy, December 2001.
- [9] P. Collet, E. Lutton, F. Raynal, and M. Schoenauer, "Polar IFS + parisian genetic programming = efficient IFS inverse problem solving," *Genetic Programming and Evolvable Machines*, vol. 1, no. 4, pp. 339–361, 2000.
- [10] Y. Davidor, *Genetic Algorithms and Robotics: A Heuristic Strategy for Optimization*, vol. 1 of *World Scientific Series in Robotics and Automated Systems*, World Scientific, Teaneck, NJ, USA, 1991.
- [11] L. Davis, *Genetic Algorithms and Simulated Annealing*, Pittman, London, UK, 1987.
- [12] T. E. Davis and J. C. Principe, "A simulated annealing like convergence theory for the simple genetic algorithm," in *Proceedings of the 4th International Conference on Genetic Algorithms (ICGA '91)*, pp. 174–181, San Diego, Calif, USA, July 1991.
- [13] M. Dorigo and G. Di Caro, "The ant colony optimization meta-heuristic," in *New Ideas in Optimization*, D. Corne, M. Dorigo, and F. Glover, Eds., pp. 11–32, McGraw-Hill, New York, NY, USA, 1999.
- [14] M. Dorigo, G. Di Caro, and L. M. Gambardella, "Ant algorithms for discrete optimization," *Artificial Life*, vol. 5, no. 2, pp. 137–172, 1999.
- [15] E. Dunn, G. Olague, and E. Lutton, "Parisian camera placement for vision metrology," *Pattern Recognition Letters*, vol. 27, no. 11, pp. 1209–1219, 2006.
- [16] A. E. Eiben and J. E. Smith, *Introduction to Evolutionary Computing*, Natural Computing Series, Springer, New York, NY, USA, 2003.
- [17] D. E. Goldberg, *Genetic Algorithms in Search, Optimization, and Machine Learning*, Addison-Wesley, Reading, Mass, USA, 1989.
- [18] D. E. Goldberg and J. Richardson, "Genetic algorithms with sharing for multimodal function optimization," in *Genetic Algorithms and Their Applications*, J. J. Grefenstette, Ed., pp. 41–49, Lawrence Erlbaum Associates, Hillsdale, NJ, USA, 1987.

- [19] D. E. Goldberg and R. E. Smith, "Nonstationary function optimization using genetic algorithm with dominance and diploidy," in *Proceedings of the 2nd International Conference on Genetic Algorithms on Genetic Algorithms and Their Application*, pp. 59–68, Lawrence Erlbaum Associates, Cambridge, Mass, USA, July 1987.
- [20] H. Hamda, F. Jouve, E. Lutton, M. Schoenauer, and M. Sebag, "Compact unstructured representations for evolutionary design," *Applied Intelligence*, vol. 16, no. 2, pp. 139–155, 2002.
- [21] A. Hill and C. J. Taylor, "Model-based image interpretation using genetic algorithms," *Image and Vision Computing*, vol. 10, no. 5, pp. 295–300, 1992.
- [22] F. Hoffmeister and T. Bäck, "Genetic algorithms and evolution strategies—similarities and differences," in *Proceedings of the 1st Workshop on Parallel Problem Solving from Nature (PPSN '90)*, vol. 496 of *Lecture Notes in Computer Science*, pp. 455–469, Dortmund, Germany, October 1991.
- [23] F. Hoffmeister and T. Baeck, "Genetic algorithms and evolution strategies: similarities and differences," Tech. Rep. SYS-1/92, University of Dortmund, Dortmund, Germany, February 1992.
- [24] J. H. Holland, "Outline for a logical theory of adaptive systems," *Journal of the Association for Computing Machinery*, vol. 9, no. 3, pp. 297–314, 1962.
- [25] J. H. Holland, *Adaptation in Natural and Artificial System*, University of Michigan Press, Ann Arbor, Mich, USA, 1975.
- [26] J. Horn, "Finite Markov chain analysis of genetic algorithms with niching," IliGAL Report 93002, University of Illinois at Urbana Champaign, Urbana, Ill, USA, February 1993.
- [27] K. A. De Jong, *Analysis of the behavior of a class of genetic adaptive systems*, Ph.D. thesis, University of Michigan, Ann Arbor, Mich, USA, 1975.
- [28] S. Kamohara, H. Takagi, and T. Takeda, "Control rule acquisition for an arm wrestling robot," in *Proceedings of the IEEE International Conference on Systems, Man and Cybernetics*, vol. 5, pp. 4227–4231, Orlando, Fla, USA, October 1997.
- [29] J. R. Koza, *Genetic Programming*, MIT Press, Cambridge, Mass, USA, 1992.
- [30] Y. Landrin-Schweitzer, P. Collet, and E. Lutton, "Interactive GP for data retrieval in medical databases," in *Proceedings of the 6th European Conference on Genetic Programming (EuroGP '03)*, vol. 2610 of *Lecture Notes in Computer Science*, pp. 93–106, Essex, UK, April 2003.
- [31] Y. Landrin-Schweitzer, P. Collet, E. Lutton, and T. Prost, "Introducing lateral thinking in search engines with interactive evolutionary algorithms," in *Proceedings of the Annual ACM Symposium on Applied Computing (SAC '03)*, pp. 214–219, Melbourne, Fla, USA, March 2003, special track on computer applications in health care.
- [32] W. B. Langdon and W. Banzhaf, "Genetic programming bloat without semantics," in *Proceedings of the 6th International Conference Parallel Problem Solving from Nature (PPSN '00)*, vol. 1917 of *Lecture Notes in Computer Science*, pp. 201–210, Paris, France, September 2000.
- [33] R. Leriche and R. T. Haftka, "Optimization of laminate stacking sequence for buckling load maximization by genetic algorithm," *AIAA Journal*, vol. 31, no. 5, pp. 951–969, 1993.
- [34] J. Lévy Véhel and E. Lutton, "Optimization of fractal functions using genetic algorithms," in *Fractals in the Natural and Applied SciencesIFIP Transactions A—Computer Science and Technology*, M. M. Novak, Ed., vol. 41, pp. 275–288, Elsevier B.V., Atlanta, Ga, USA, 1993.
- [35] J. Lévy Véhel and E. Lutton, "Evolutionary signal enhancement based on Hölder regularity analysis," in *Proceedings of Applications of Evolutionary Computing: EvoWorkshops: EvoCOP, EvoFlight, EvoIASP, EvoLearn, and EvoSTIM*, E. J. W. Boers, J. Gottlieb, P. L. Lanzi, et al., Eds., vol. 2037 of *Lecture Notes in Computer Science*, pp. 325–334, Como, Italy, April 2001.
- [36] J. Louchet, M. Guyon, M.-J. Lesot, and A. Boumaza, "Dynamic flies: a new pattern recognition tool applied to stereo sequence processing," *Pattern Recognition Letters*, vol. 23, no. 1–3, pp. 335–345, 2002.
- [37] E. Lutton, P. Collet, and J. Louchet, "EASEA comparisons on test functions: GAlib versus EO," in *Proceedings of the 5th International Conference on Evolution Artificielle (EA '01)*, P. Collet, C. Fonlupt, J.-K. Hao, E. Lutton, and M. Schoenauer, Eds., vol. 2310 of *Lecture Notes in Computer Science*, pp. 219–230, Springer, Le Creusot, France, October 2002.

- [38] Z. Michalewicz, *Genetic Algorithms + Data Structures = Evolution Programs*, Springer, New York, NY, USA, 1992.
- [39] N. Monmarche, G. Nocent, G. Venturini, and P. Santini, “On generating HTML style sheets with an interactive genetic algorithm based on gene frequencies,” in *Proceedings of the 4th European Conference on Artificial Evolution (AE '99)*, C. Fonlupt, J. K. Hao, E. Lutton, M. Schoenauer, and E. Ronald, Eds., vol. 1829 of *Lecture Notes in Computer Science*, pp. 99–110, Springer, Dunkerque, France, November 1999.
- [40] A. E. Nix and M. D. Vose, “Modeling genetic algorithms with Markov chains,” *Annals of Mathematics and Artificial Intelligence*, vol. 5, no. 1, pp. 79–88, 1992.
- [41] G. Olague and C. Puent, “Parisian evolution with honeybees for three-dimensional reconstruction,” in *Proceedings of the 8th Annual Genetic and Evolutionary Computation Conference (GECCO '06)*, vol. 1, pp. 191–198, Seattle, Wash, USA, July 2006.
- [42] M. A. Potter and K. A. De Jong, “Cooperative coevolution: an architecture for evolving coadapted subcomponents,” *Evolutionary Computation*, vol. 8, no. 1, pp. 1–29, 2000.
- [43] G. Roth and M. D. Levine, “Geometric primitive extraction using a genetic algorithm,” in *Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR '92)*, pp. 640–643, Champaign, Ill, USA, June 1992.
- [44] F. Rothlauf, J. Branke, S. Cagnoni, et al., Eds., “Applications of Evolutionary Computing, EvoWorkshops: EvoBIO, EvoCOMNET, EvoHOT, EvoIASP, EvoINTERACTION, EvoMUSART, and EvoSTOC,” vol. 3907 of *Lecture Notes in Computer Science*, Springer, Budapest, Hungary, April 2006.
- [45] H.-P. Schwefel, *Numerical Optimization of Computer Models*, John Wiley & Sons, New York, NY, USA, 2nd edition, 1995.
- [46] K. Sims, “Interactive evolution of dynamical systems,” in *Proceedings of the 1st European Conference on Artificial Life (ECAL '91)*, pp. 171–178, Paris, France, 1991.
- [47] K. Sims, “Artificial evolution for computer graphics,” *Computer Graphics*, vol. 25, no. 4, pp. 319–328, 1991.
- [48] T. Soule, J. A. Foster, and J. Dickinson, “Code growth in genetic programming,” in *Proceedings of the 1st Annual Conference Genetic Programming*, J. R. Koza, D. E. Goldberg, D. B. Fogel, and R. L. Riolo, Eds., pp. 215–223, MIT Press, Cambridge, Mass, USA, July 1996.
- [49] H. Takagi, “Interactive evolutionary computation: system optimisation based on human subjective evaluation,” in *Proceedings of IEEE International Conference on Intelligent Engineering Systems (INES '98)*, pp. 1–6, Vienna, Austria, 1998.
- [50] H. Takagi and M. Ohsaki, “IEC-based hearing aid fitting,” in *Proceedings of IEEE International Conference on System, Man and Cybernetics (SMC '99)*, vol. 3, pp. 657–662, Tokyo, Japan, October 1999.
- [51] S. J. P. Todd and W. Latham, *Evolutionary Art and Computers*, Academic Press, Amsterdam, The Netherlands, 1992.
- [52] P. Trompette, J. L. Marcellin, and C. Schmeling, “Optimal damping of viscoelastic constrained beams or plates by use of a genetic algorithm,” in *Proceedings of International Union of Theoretical and Applied Mechanics (IUTAM '93)*, Zakopane, Poland, August-September 1993.
- [53] L. Trujillo and G. Olague, “Synthesis of interest point detectors through genetic programming,” in *Proceedings of the 8th Annual Conference on Genetic and Evolutionary Computation (GECCO '06)*, pp. 887–894, Seattle, Wash, USA, July 2006.
- [54] S. Truvé, “Using a genetic algorithm to solve constraint satisfaction problems generated by an image interpreter,” in *Proceedings of the 7th Scandinavian Conference on Image Analysis on Theory and Applications of Image Analysis*, pp. 378–386, Aalborg, Denmark, August 1991.
- [55] M. Vose, “Modeling simple genetic algorithms,” in *Proceedings of Foundations of Genetic Algorithms (FOGA '92)*, pp. 24–29, Vail, Colo, USA, July 1992.

- [56] E. Zitzler, K. Deb, and L. Thiele, “Comparison of multiobjective evolutionary algorithms: empirical results,” *Evolutionary Computation*, vol. 8, no. 2, pp. 173–195, 2000.

Stefano Cagnoni: Dipartimento di Ingegneria dell'Informazione, Università di Parma,
Viale Usberti 181a, 43100 Parma, Italy

Email: cagnoni@ce.unipr.it

Evelyne Lutton: APIS team, INRIA Futurs, Parc Orsay Università, 4 rue Jacques Monod,
91893 Orsay Cedex, France

Email: evelyne.lutton@inria.fr

Gustavo Olague: EvoVision Project, Computer Science Department, CICESE Research Center,
Km. 107 Carretera Tijuana-Ensenada, 22860 Ensenada, Mexico

Email: olague@cicese.mx

2

Evolutionary multifractal signal/image denoising

Evelyne Lutton and Jacques Lévy Véhel

This chapter investigates the use of evolutionary techniques for multifractal signal/image denoising. Two strategies are considered: using evolution as a pure stochastic optimizer, or using interactive evolution for a meta-optimization task. Both strategies are complementary as they allow to address different aspects of signal/image denoising.

2.1. Introduction

We deal with enhancement, or denoising, of complex signals, based on the analysis of the local Hölder regularity. Our methods do not make explicit assumptions on the type of noise nor on the global smoothness of original data, but rather suppose that signal enhancement is equivalent to increasing the Hölder regularity at each point. Such methods are well adapted to the case where the signal to be recovered is itself very irregular, for example, nowhere differentiable with rapidly varying local regularity.

We describe two techniques. The first one tries to find a signal close to the observations and such that its *local Hölder function* is prescribed. A pure optimization approach is convenient in this case, as this problem does not admit a closed-form solution in general (although attempts have been previously done on an analytical basis for simplified cases [17, 19]). In addition, the number of variables involved is huge. Genetic algorithms have been found to be efficient in this case, and yield better results than other algorithms. The principles and example results are presented in Section 2.2.

However, it appears that the question of results evaluation is critical: a precise (and general) definition of what good denoising, or enhancement, is, is questionable. Medical doctors indeed may have different opinions on the quality of a given result, as well as remote sensing specialists, or art photographers. The perception of quality is extremely dependent on the end-user, the context, and the type of application. A simple signal-to-noise ratio (when computable) is certainly not able to capture the subtle perceptive judgment of a human end-user.

A print edition of this book can be purchased at

<http://www.hindawi.com/spc.8.html>

<http://www.amazon.com/dp/9774540018>

To investigate this issue, we describe another regularity-based enhancement technique: multifractal Bayesian denoising acts by finding a signal close to the observations and such that its *multifractal spectrum* is prescribed. This method relies on the tuning of a small set of parameters that are able to provide various improvements of the observed noisy image. An interactive evolutionary approach has been designed in order to cope with the meta-optimization problem of tuning the parameters set and is described in Section 2.7.

In order to get acceptable computation times, the underlying optimization problem and its parameters have been designed to be solved by a deterministic method. The evolutionary approach is used in an interactive way, at a meta-level.

Going further into this direction, a scheme has been designed (and tested) in order to reduce the number of user interactions, in other words to limit the famous “user fatigue,” see Section 2.9.

The schemes and tools developed on the signal and image denoising problem can be extended to other image-analysis tasks, such as multifractal image segmentation (see Section 2.11).

2.2. Signal enhancement/denoising

The problem may be set in the following way: someone observes a signal Y which is a certain combination $F(X, B)$ of the signal of interest X and a “noise” B . Making various assumptions on the noise, the structure of X and the function F , one then tries to derive a method to obtain an estimate \hat{X} of the original signal which is optimal in some sense. Most commonly, B is assumed to be independent of X , and, in the simplest case, is taken to be white, Gaussian, and centred. F usually amounts to convoluting X with a lowpass filter and adding the noise. Assumptions on X are related to its regularity, for example, X is supposed to be piecewise C^n for some $n \geq 1$. Techniques proposed in this setting resort to two domains: functional analysis and statistical theory. In particular, wavelet-based approaches, developed in the last ten years, may be considered from both points of view [7, 8].

In this work, we do not make explicit assumptions on the type of noise and the coupling between X and B through F . Furthermore, we are not interested in the global smoothness of X , but rather concentrate on its *local* regularity: enhancement will be performed by increasing the Hölder function α_Y (see next section for definitions) of the observations. Indeed, it is generally true that the local regularity of the noisy observations is smaller than the one of the original signal, so that, in any case, $\alpha_{\hat{X}}$ should be greater than α_Y . We thus define our estimate \hat{X} to be the signal “closest” to the observations which has the desired Hölder function. Note that since the Hölder exponent is a local notion, this procedure is naturally adapted for signals which have sudden changes in regularity, like discontinuities. From a broader perspective, such a scheme is appropriate when one tries to recover signals which are highly irregular and for which it is important that the denoising procedure yields the right regularity structure (i.e., preserves the evolution of α_X along the path).

2.3. The local Hölder exponent

We will measure the local irregularity of signals with the help of the Hölder exponent. To simplify notations, we assume that our signals are nowhere differentiable. Generalization to other signals require technicalities which are unessential to our purposes.

Let $\alpha \in (0, 1)$, $\Omega \subset \mathbf{R}$. One says that $f \in C_l^\alpha(\Omega)$ if

$$\exists C : \forall x, y \in \Omega : \frac{|f(x) - f(y)|}{|x - y|^\alpha} \leq C. \quad (2.1)$$

Let $\alpha_l(f, x_0, \rho) = \sup\{\alpha : f \in C_l^\alpha(B(x_0, \rho))\}$. Note that $\alpha_l(f, x_0, \rho)$ is nonincreasing as a function of ρ .

We are now in position to give the definition of the local Hölder exponent.

Definition 2.1. Let f be a continuous function. The local Hölder exponent of f at x_0 is the number $\alpha_l(f, x_0) = \lim_{\rho \rightarrow 0} \alpha_l(f, x_0, \rho)$.

Since α_l is defined at each point, we may associate to f the function $x \rightarrow \alpha_l(x)$ which measures the evolution of its regularity.

This regularity characterization is widely used in fractal analysis because it has direct interpretations both mathematically and in applications. For instance, the computation of the Hölder exponent at each point of an image allows to perform edge detection [6].

2.4. Signal enhancement based on increasing the local Hölder function

Let X denote the original signal and Y the degraded observations. We seek a regularized version \hat{X} of Y that meets the following constraints: (a) \hat{X} is close to Y in the L^2 sense, (b) the (local) Hölder function of \hat{X} is prescribed.

If α_X is known, we choose $\alpha_{\hat{X}} = \alpha_X$. In some situations, α_X is not known but can be estimated from Y . Otherwise, we just set $\alpha_{\hat{X}} = \alpha_Y + \delta$, where δ is a user-defined positive function, so that the regularity of \hat{X} will be everywhere larger than the one of the observations. We must solve two problems in order to obtain \hat{X} . First, we need a procedure that estimates the local Hölder function of a signal from discrete observations. Second, we need to be able to manipulate the data so as to impose a specific regularity.

We will use a wavelet-based procedure for estimating and controlling the Hölder function. We let $\{\psi_{j,k}\}_{j,k}$ be an orthonormal wavelet basis, where as usual j represents scale and k position. Denote $c_{j,k}$ the wavelet coefficient of X . Results in [10, 11] indicate that, assuming that ψ is regular enough and has sufficiently many vanishing moments, one may estimate $\alpha_X(t)$ by linear regression of $\log(|c_{j,k}|)$ with respect to the scale j (\log denotes base 2 logarithm), considering those indices (j, k) such that the support of $\psi_{j,k}$ is centred above t . Roughly speaking,

those coefficients should decay in scale as $2^{-j(\alpha+1/2)}$ (more precisely, all the $|c_{j,k}|$ are bounded by $C2^{-j(\alpha+1/2)}$ for some $C > 0$, and some of the coefficients $|c_{j,k}|$ are of the order of $C2^{-j(\alpha+1/2)}$).

The use of wavelets then allows to perform the reconstruction in a simple way. Starting from the coefficient $(d_{j,k})$ of the observations, we will define a procedure that modifies them to obtain coefficients $(c_{j,k})$ that fulfil the decay condition with the desired α , and then reconstruct \hat{X} from those $(c_{j,k})$.

We may now reformulate our problem as follows: for a given set of observations $Y = (Y_1, \dots, Y_{2^n})$ and a target Hölder function α , find \hat{X} such that $\|\hat{X} - Y\|_{L^2}$ is minimum and the regression of the logarithm of the wavelet coefficients of \hat{X} above any point i with respect to scale is $-(\alpha(i) + 1/2)$. Note that we must adjust the wavelet coefficients in a global way. Indeed, each coefficient at scale j subsumes information about roughly 2^{n-j} points. Thus, we cannot consider each point i sequentially and modify the wavelet coefficients above it to obtain the right regularity, because point $i+1$, which shares many coefficients with i , requires different modifications. The right way to control the regularity is to write the regression constraints simultaneously for all points. This yields a system which is linear in the logarithm of the coefficients:

$$\Delta L = A, \quad (2.2)$$

where Δ is a $(2^n, 2^{n+1} - 1)$ matrix of rank 2^n , and

$$\begin{aligned} L &= (\log |c_{1,1}|, \log |c_{2,1}|, \log |c_{2,2}|, \dots, \log |c_{n,2^n}|), \\ A &= -\frac{n(n-1)(n+1)}{12} \left(\alpha(1) + \frac{1}{2}, \dots, \alpha(2^n) + \frac{1}{2} \right). \end{aligned} \quad (2.3)$$

Since we use an orthonormal wavelet basis, the requirements on the $(c_{j,k})$ may finally be written as follows:

$$\text{minimize} \sum_{j,k} (d_{j,k} - c_{j,k})^2 \text{ subject to, } \forall i = 1, \dots, 2^n, \quad (2.4)$$

$$\sum_{j=1}^n s_j \log (|c_{j,E((i-1)2^{j+1-n})}|) = -M_n \left(\alpha(i) + \frac{1}{2} \right), \quad (2.5)$$

where $E(x)$ denotes the integer part of x and the coefficients $s_j = j - (n+1)/2$, $M_n = (n(n-1)(n+1))/12$, and (2.5) is deduced from the requirement that the linear regression of the wavelet coefficients of \hat{X} above position i should be $-(\alpha(i) + 1/2)$.

Finding the global solution to the above program is a difficult task; in particular, it is not possible to find a closed form formula for the $c_{j,k}$. In [19], a method

is described, that allows explicit computations under simplifying assumptions. In the following, we show how this problem can be addressed with an evolutionary algorithm.

2.5. Evolutionary signal enhancement with EASEA

An evolutionary technique seems to be appropriate for the optimization problem described in (2.4): a large number of variables are involved, and the function to be optimized as well as the constraint is nonlinear. We describe in this section an implementation based on the EASEA [5] language and compiler.

EASEA (EAsy Specification of Evolutionary Algorithms) is a language dedicated to evolutionary algorithms. Its aim is to relieve the programmer of the task of learning how to use evolutionary libraries and object-oriented programming by using the contents of a user-written `.ez` source file.

EASEA source files only need to contain the “interesting” parts of an evolutionary language, namely the fitness function, specification of the crossover and mutation operators, the initialization of a genome plus a set of parameters describing the run. With this information, the EASEA compiler creates a complete C++ source file containing function calls to an evolutionary algorithms library (either the GALIB or EO for EASEA v0.6). Therefore, the minimum requirement necessary to write evolutionary algorithms is the capability of creating non-object-oriented functions, specific to the problem which needs to be solved.

In our case, the evolutionary optimization involved to enhance a signal (1D or 2D) was implemented using a simple structure on which genetic operators were defined. We used GALib [35] as the underlying evolutionary library.

We describe below the implementation for 1D signals. An implementation for image denoising was also produced based on the same principles [23].

The Haar wavelet transform has been used to produce the $d_{j,k}$ associated to the observed signal Y . We also suppose that we know the desired Hölder exponents $\alpha(i)$ (either $\alpha(i) = \alpha_Y(i) + \delta$ where the $\alpha_Y(i)$ are the Hölder exponents of Y and δ is a user-defined regularization factor, or $\alpha(i)$ is set a priori).

Our unknowns will be the multiplicative factors $u_{j,k}$ such that $c_{j,k} = u_{j,k} * d_{j,k}$, $j \in [0 \dots n - 1]$, $k \in [0, \dots, 2^j - 1]$. As is usual in wavelet denoising, we leave unchanged the first l levels and seek for the remaining $u_{j,k}$ in $[0, 1]$. The genome is made of the $u_{j,k}$ coefficients, for $j \in [l, \dots, n - 1]$ and $k \in [0, \dots, 2^j - 1]$. These coefficients are encoded as a real numbers vector of size $\text{SIZE_MAX} = 2^n - 2^l$, which can be written using EASEA syntax as

```
GenomeClass { double U[SIZE_MAX]; }
```

The EASEA Standard functions sections contain the specific genetic operators, namely the following.

- (1) *The initialization function:* each $u_{j,k}$ coefficient is randomly set to a value in $[0, 1]$. Two initial solutions are also put in the initial population: $u_{j,k} = 1$, and $u_{j,k} = 2^{-k\delta}$.
- (2) *The crossover function:* a barycentric crossover has been easily defined as follows: let `parent1` and `parent2` be the two genomes out of which

child1 and child2 must be generated, and let alpha be a random factor:

```
\GenomeClass::crossover:
    double alpha = (double)random(0.,1.);
    if (&child1) for (int i=0; i<SIZE_MAX; i++)
        child1.U[i] = alpha*parent1.U[i]
                    + (1.-alpha)*parent2.U[i];
    if (&child2) for (int i=0; i<SIZE_MAX; i++)
        child2.U[i] = alpha*parent2.U[i]
                    + (1.-alpha)*parent1.U[i];
\end
```

- (3) *The mutation function:* mutation is a random perturbation of radius $SIGMA = 0.01$, applied with probability PMut on each gene.

```
\GenomeClass::mutator: // Must return the number of
                        // mutations as an int
    int NbMut=0;
    for (int i=0; i<SIZE_MAX; i++)
        if (tossCoin(PMut)){
            NbMut++;
            Genome.U[i]+=SIGMA*(double)random(-1.,1.);
            Genome.U[i] = MIN(1.,Genome.U[i]);
            Genome.U[i] = MAX(0.,Genome.U[i]);}
    if (NbMut==0) identicalGenome=true; // saves evaluation
                                                // time
    return NbMut;
\end
```

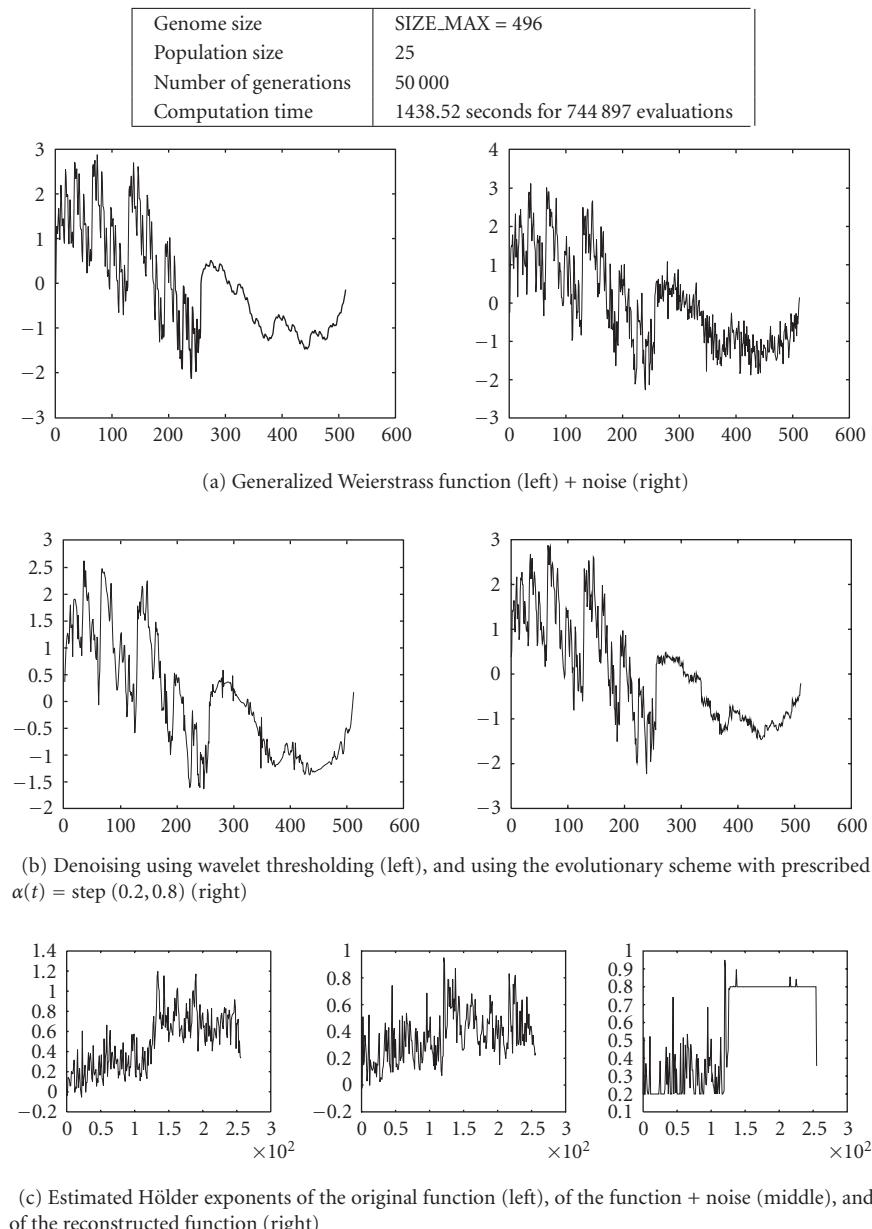
- (4) *The evaluation function:* the fitness function has two aims: minimize $\sum((1 - u_{j,k}) * c_{j,k})^2$, making sure constraint (2.4) is satisfied, that is, the Hölder exponents are the ones we want. The constraint is integrated into the fitness function using a high-penalty factor W :

$$\text{Fitness} = \sum_{j,k} ((1 - u_{j,k}) * c_{j,k})^2 + W * \sum_i |\alpha_u(i) - \alpha(i)|. \quad (2.6)$$

We use the GALib steady state genetic engine with replacement percentage of 60% and selection by ranking. Crossover and mutation probabilities are fixed, respectively, to 0.9 and 0.001. Genome size, population size, and number of generations are fixed for each experiment, see Section 2.6.

2.6. Numerical experiments

Results of enhancement on synthetic 1D data are shown in Figure 2.1. The original signal is a generalized Weierstrass function [6] with $\alpha_X(t) = 0.2$ for $t < 0.5$, $\alpha_X(t) = 0.8$ for $t > 0.5$ that has been corrupted by additive white Gaussian noise. Figure 2.1 shows the original signal, the noisy one, and the result of the enhancement procedure. For comparison, a denoising using a classical wavelet hard thresholding is also displayed (i.e., all coefficients with absolute value smaller than a given

FIGURE 2.1. Results on a generalized Weierstrass function $\alpha(t) = \text{step}(0.2, 0.8)$.

threshold are put to 0). For both procedures, the parameters were set so as to obtain the best fit to the known original signal. It is seen that, for such irregular signals, the Hölder regularity-based enhancement yields more satisfactory results.

One should however remark that we have placed ourselves in a favorable situation for the evolutionary algorithm, since the constraint was to find the (generally unknown) correct Hölder function. Parameters of the evolutionary algorithm are given in Figure 2.1.

2.7. Interactive schemes and multifractal Bayesian denoising

There are several ways to improve the method. To begin with, more precise estimations of Hölder exponents yield more accurate results. For instance, in [21], an estimation based on the analysis of local oscillations of the signal has been used. The associated inverse problem is then more complex, and necessitates the use of specific genetic operators. Improved results are obtained at the expense of more complex computations.

Progress on this topic is however more related to the design of an efficient analysis of the quality of the results. But computational measurement of denoising results is difficult to design, as the evaluation of a good denoising is strongly dependent on the end-user as well as the application framework. Signal-to-noise ratio is unable to reflect all the subtle components of a human expert appreciation on a denoising result. Remote sensing, medical imaging, sound restoration, data filtering have very different constraints, and expert users of these domains have different needs.

A way to cope with this discrepancy is to involve the human user in the optimization loop in order to let him accurately guide the search mechanism towards what he wishes. The artificial evolution framework allows one to introduce human evaluation in the algorithmic loop, and to cope with human judgment irregularity (or even inconsistency). Actually, interactive evolution is a research topic that is rapidly growing: first attempts were oriented toward artistic applications [1, 30, 31, 34], but now many other applications domains are explored: hearing aids fitting [33], smooth, human-like, control rules design for a robot arm [13], or design of HTML style sheets [25]. An overview of this vast topic can be found in [32].

Interaction with humans raises several problems mainly related to human fatigue. Three types of solutions have been considered [2, 27, 32]: (1) reduce the size of the population and the number of generations, (2) choose specific models to constrain the research in a priori “interesting” areas of the search space, or (3) perform automatic learning (based on a limited number of characteristic quantities) in order to assist the user and only display the most interesting individuals in the population, with respect to previous votes of the user.

In order to implement (1) and (2), we adopt an approach different from the one in the previous sections. Instead of prescribing the Hölder exponent at each point, we will rather try to control the multifractal spectrum of the denoised image. This allows to reduce dramatically the number of variables. A first experiment where a small population is evolved using a multifractal scheme is presented in Section 2.8. We then experimented an approach integrating item (3), that is, we

extend fitness rating to individuals in a larger population via the analysis of the user judgment on a small sample of individuals (Section 2.9).

2.7.1. Description of the multifractal Bayesian denoising method

The multifractal analysis of a signal consists in measuring its regularity at each sample point, in grouping the points having the same irregularity, and then in estimating the “size” (through some “fractal dimension”) of each iso-regularity set. Irregularity is measured via the Hölder exponent.

The multifractal spectrum f is a representation of the irregularity of the signal over its definition domain. For each irregularity value, that is, for each α , one estimates the speed of exponential decay of the probability of finding a point with regularity α as resolution tends to infinity. In some cases, this speed is also the Hausdorff dimension of the corresponding iso- α set (see [3] for details).

For example, for an image, a value of $f(\alpha) \simeq 1$ corresponds to a set of points with same regularity and dimension 1 (i.e., it will most of the times look like a set of lines), $f(\alpha) \simeq 0$ is a set of scattered points (singular points), and $f(\alpha) \simeq 2$ is a typically an area of positive measure.

Multifractal analysis is a tool widely used in image and signal analysis, as it provides at the same time a local (α) and a global ($f(\alpha)$) viewpoints on data. It has been exploited with success in many applications where irregularity bears some important informations (image segmentation [15], signal and image denoising [16, 20], etc.)

The principle of the denoising method is the following, for a noisy image I_1 , we search for a denoised image I_2 that satisfies two conditions:

- (i) I_2 has a given multifractal spectrum;
- (ii) the probability that the addition of a white Gaussian noise (with variance σ) to I_2 produces the observed image I_1 is maximal.

As we mentioned before, the wavelet transform is a convenient tool for the estimation of the Hölder exponents. This second denoising method is thus also based on the discrete wavelet transform.

2.7.2. The search space is the set of free parameters of the method

We explain here the algorithm for image denoising. Recall that the aim is to find a denoised image I_2 close to the noisy observations I_1 , under the constraint that I_2 has a given multifractal spectrum g . The noise is assumed to be white, centred, and Gaussian with variance σ .

The problem may be reformulated as follows: if we denote by y a wavelet coefficient of the noisy image at scale j , then the corresponding wavelet coefficient \hat{x} of the denoised image at the same scale j can be calculated by solving the following equation (for details, see [18]):

$$\hat{x} = \arg \max_{x>0} \left(j \cdot g \left(\frac{\log_2(\hat{K} \cdot x)}{-j} \right) - \frac{(|y| - x)^2}{2\sigma^2} \right) \operatorname{sgn}(y), \quad (2.7)$$

where

- (i) \hat{K} is a constant for which $\hat{K} \cdot |y| < 1$ holds and may be set independently for each scale. In what follows, \hat{K} has been taken as the inverse of the maximum coefficient in each scale j ;
- (ii) g is a function which defines the multifractal spectrum of the denoised image. If we choose to represent it by a linear-by-parts function, its shape is determined by 5 values α_{\min} , α_{nod} , α_{\max} , $g(\alpha_{\min})$, and $g(\alpha_{\max})$. More precisely, the spectrum has been chosen to fulfil the following constraints:
 - (a) g is defined on the interval $[\alpha_{\min}, \alpha_{\max}]$,
 - (b) $g(x) \in [0, 1]$,
 - (c) $\alpha_{\text{nod}} \in [\alpha_{\min}, \alpha_{\max}]$ and $g(\alpha_{\text{nod}}) = 1$,
 - (d) g is affine on $[\alpha_{\min}; \alpha_{\text{nod}}]$ and on $[\alpha_{\text{nod}}; \alpha_{\max}]$.

In most cases, but not necessarily, the multifractal spectrum calculated from the denoised coefficients \hat{x} should show a slight spectral shift to the right. This shift is a sign of an overall increase of regularity.

Consequently, we have 7 free parameters:

- (i) the 5 values defining the a priori spectrum g ,
- (ii) the variance σ of the noise,
- (iii) the wavelet used for the discrete (inverse) wavelet transformation.

The choice of the wavelet is less critical than the choice of the other 6 parameters. Usually, Daubechies 6 to 12 offer equivalent denoising results in terms of visual reception whereas Daubechies wavelets with smaller supports yield unsatisfactory results in some cases.

Especially in cases where we want to treat very noisy images and subsequently have to set the parameters σ and α_{nod} to relatively high values, the denoising algorithm leads to artefacts in the denoised image when using wavelets with a small support, see Figure 2.2. The regularity of those wavelets is low. They are therefore not able to model very irregular parts of an image.

It should be mentioned that the number of calculated wavelet scales is fixed to a value obtained from the image dimensions $[N \times M]$:

$$\text{scales} = \lfloor \log 2(\max(N, M)) \rfloor. \quad (2.8)$$

The setup of the 7 resulting free parameters is nontrivial in the sense that they are strongly dependent on the amount of noise in the noisy image and the subjective opinion of the human observer about which result reflects best the desired denoised image.

A solution is therefore to build an interactive evolutionary algorithm (IEA) to interactively find suitable settings of the free parameters.

2.8. An interactive approach with a small population

This first implementation does not include the choice of the wavelet basis as a free parameter but considers a shift to the a priori spectrum g for diagonal wavelet

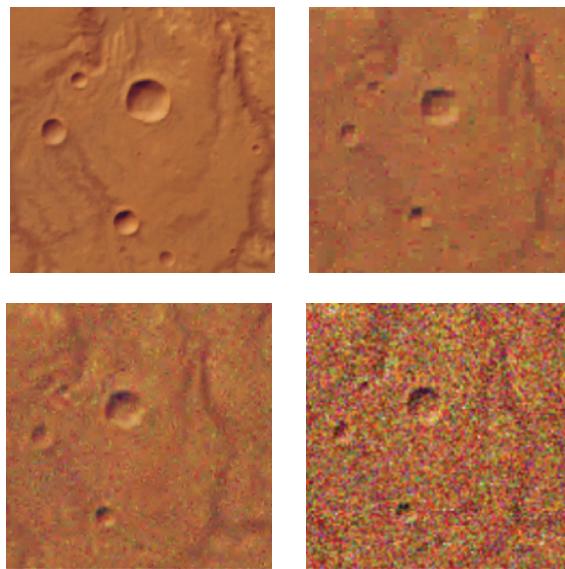


FIGURE 2.2. *Original image without noise* (up left), multifractal denoising using wavelet *Daubechies 2* (up right), multifractal denoising using wavelet *Daubechies 18* (low left), *noisy image* (low right); all parameters except wavelets are constant.

coefficients. It has been noticed that the diagonal wavelet coefficients are more sensitive to additive noise and therefore may need a different spectrum g .

The genomes that are evolved by this IEA are made of 7 real genes:

- (i) 5 values to define the g function for the horizontal and vertical wavelet coefficients: $\alpha_{\min} \in [0, 0.5]$, $g(\alpha_{\min}) \in [0, 1]$, $\alpha_{\text{nod}} > \alpha_{\min}$, $\alpha_{\text{nod}} \in [0, 2]$, $\alpha_{\max} > \alpha_{\text{nod}}$, $\alpha_{\max} \in [0.01, 20]$, $g(\alpha_{\max}) \in [0.2, 1]$;
- (ii) the shift of the g function for the diagonal coefficients (range $[0, 0.5]$);
- (iii) the variance of Gaussian noise, σ (range $[0, 100.0]$).

Fitness and user interaction. The fitness function is given by the user via sliders attached to each denoised sample. Evaluations range from -10 to $+10$. The default value 0 corresponds to a neutral judgment.

Additionally, the user may directly edit the genotypes of each image, see Figure 2.3, and thus participate in the evolutionary loop as an additional genetic operator.

Genetic engine. The population has a fixed size of 6 individuals. Each individual carries a set of 7 parameters and therefore represents a potential solution for the aforementioned optimization problem. All individuals are presented as an image, resulting from the denoising algorithm with corresponding free parameters. The basic evolutionary cycle is presented in Figure 2.4, and the operators are the following.

- (i) *Parents selection* is performed by deterministic selection of the 3 best individuals in the population.

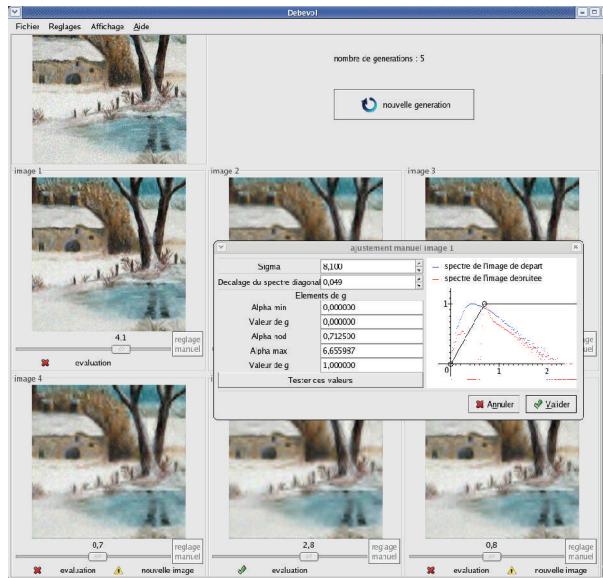


FIGURE 2.3. The interface of the small population IEA, written in C++.

(ii) *Genetic operators:*

- (a) *barycentric crossover* is performed by weighted combination of parents with a randomly chosen weight in $[0, 1]$;
- (b) *mutation* is an independent perturbation of each gene value by addition of a Gaussian noise with a given variance.

(iii) *Survivor selection* scheme replaces the 3 worst parents by offsprings.

A sharing scheme is applied before parent selection: the user marks are weighted to maintain diversity inside the small population. The sharing is based on a genotypic distance. The parent selection then chooses the 3 individuals with the best weighted fitness and is therefore fully deterministic. Crossover and mutation operators then produce 3 children. The survivor selection substitutes parents with offspring and thereby closes the evolutionary cycle.

2.9. An interactive approach with a large population

In the previous interactive scheme, the user has access to 6 individuals (or images) per generation, and the genetic engine only considers the current user evaluations to calculate the next generation. This IEA is driven by a fitness sample, or let us say a *fitness map*, made of only 6 points.

In order to increase the reactivity of the system while being able to handle populations of any size (this increases the search capabilities), we consider the use of a larger fitness sample, while considering techniques to approximate user evaluation. However, a dynamic approximation of the interactive fitness is a delicate task, and necessitates rather large samples. We have proposed a method based on

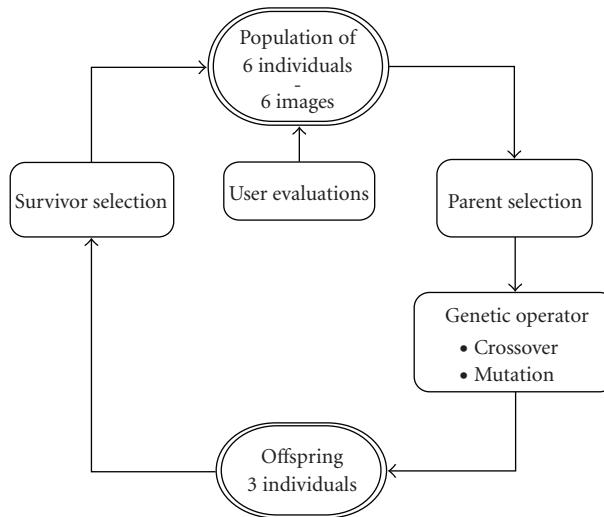


FIGURE 2.4. The small population denoising IEA genetic engine.

the use of past user marks, collected in a set, the *fitness map*. The fitness of new individuals produced by the genetic engine can be preliminarily estimated from the fitness map by smooth interpolation (flat or polynomial, see below). This preliminary fitness estimation can serve as a preselection tool in order to show to the user only the 6 best individuals of a larger current population.

The use of larger population sizes offers some major advantages, of which an obviously easier maintenance of diversity, a more extensive exploration of the given search space, and a possible speedup of convergence are the most significant.

Genome. The genomes are made of 7 genes:

- (i) 5 values to define the g function for the horizontal and vertical wavelet coefficients: $\alpha_{\min} \in [0, 0.5]$, $g(\alpha_{\min}) \in [0, 1]$, $\alpha_{\text{nod}} > \alpha_{\min}$, $\alpha_{\text{nod}} \in [0, 2]$, $\alpha_{\max} > \alpha_{\text{nod}}$, $\alpha_{\max} \in [0.01, 20]$, $g(\alpha_{\max}) \in [0.2, 1]$;
- (ii) the wavelet used for the discrete wavelet transformation (Daubechies 2 to 20);
- (iii) the variance of Gaussian noise, $\sigma \in [0, 100]$.

Fitness and user interaction. The user evaluations are given the same way as in Section 2.8 with range

$[-6 \text{ (very bad)}, \dots, 0 \text{ (neutral)}, \dots, +6 \text{ (very good)}]$.

The genetic engine is highly customizable for parameters setting. In contrast to the small population IEA, it is possible that all 6 images in the user interface are changed from a generation to the next. As loosing good images may be frustrating for the user, it is possible to mark images as “super individuals” that remain in the user interface and in the population. The user may toggle this state at any time; see Figure 2.6.

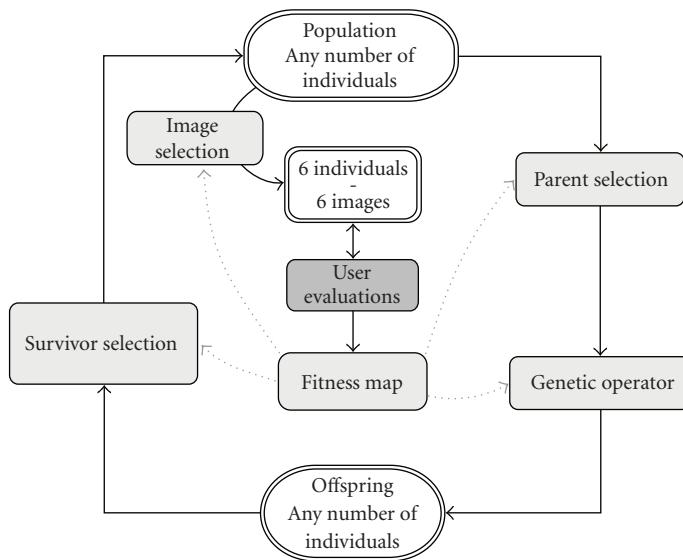


FIGURE 2.5. *The extended genetic engine supports a fitness map.*

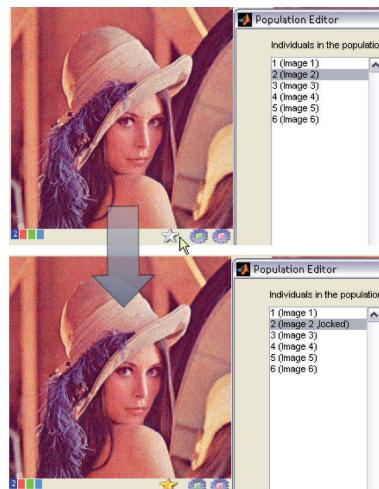


FIGURE 2.6. Clicking the star button toggles an individual as “super individual.” From thereon it is “immortal” in the population.

Additionally, to increase user interactivity, two new dialogues have been created: to view and manipulate the individuals in the population (Figure 2.7), as well as the samples in the fitness map (Figure 2.8). These dialogues both provide plots of the gene values of the individuals in the current population, or in the fitness map. By toggling checkboxes, additional curves, such as an interpolation of fitness and sharing values, are available.

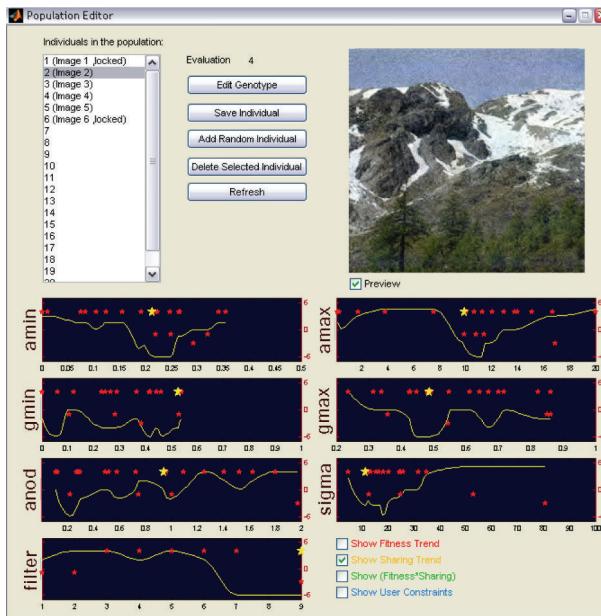


FIGURE 2.7. *The population editor*. Individuals may be added to the population, existing individuals may be deleted and their genotype can be manipulated. The gene values are plotted on 7 curves as small stars. The plotted curves are interpolations of the fitness (or shared fitness) samples values.

“User ranges” (Figure 2.9) have been introduced, as soft thresholds that constrain the search space of genes, and can be set independently for each of the 7 genes.

Genetic engine. Enlarging the population size and using a fitness map requires major changes: the selection step now strongly depends on the fitness map, as well as crossover and mutation operators. The generation cycle includes an “image selection” step, that is, 6 individuals are selected to be shown to the user, see Figure 2.5.

The fitness map is a matrix of size $[8 \times N]$. N is the number of samples that are saved in the fitness map. These samples are vectors of size $[8 \times 1]$, and include a genotype and its corresponding fitness value. The fitness map is used to interpolate between the available samples in order to predict the fitness values of unknown genotypes. Two interpolation methods have been implemented:

- (i) “Nearest”: the fitness value of the nearest sample in the fitness map is returned as the fitness value of the unknown sample.
- (ii) “Interpolation”: interpolating polynomials of order 8 are calculated for each gene using the samples of the fitness map (small stars interpolated by smooth curves in Figure 2.10). The approximated fitness value for an unknown sample (see vertical markers in Figure 2.10) is the mean value of the 7 polynomials for the genes values of the unknown sample.

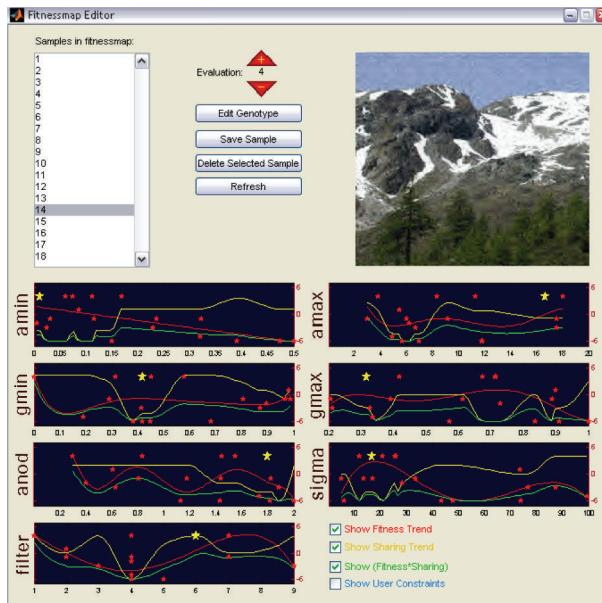


FIGURE 2.8. The fitness map editor. Samples may be deleted and their fitness can be reevaluated.

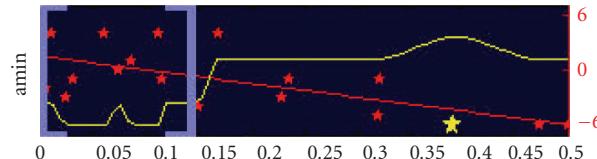


FIGURE 2.9. Plot of sample fitness for the values of α_{\min} . An interpolation of the fitness values is plotted (line). The sharing estimation is plotted as a curve. Setting a user preferred range for individual genes is done by drag and drop of the thick vertical brackets.

Various selection algorithms have been implemented. These selection operators can be deployed by the parent, offspring, and image selection. The selection operator that is actually used in a certain stage of the genetic cycle is set offline with help of a configuration file.

The available selection methods are the following.

- (i) “*Fittest*”: the individual with the best fitness value is selected.
- (ii) “*Cycle*”: n individuals are selected by cycling through m individuals among the fittest. This method can be used to generate an offspring from a small number of parent individuals (as in the small population IEA).
- (iii) “*Roulette*”: randomized variant of fitness-proportionate selection.
- (iv) “*Rank*”: randomized variant of rank-proportionate selection. The selection probability for an individual is $\text{pressure}^{-\text{rank}}$, where “*pressure*” adjusts the strength of selection and “*rank*” is the position of the individual inside the population (sorted by decreasing fitness values).

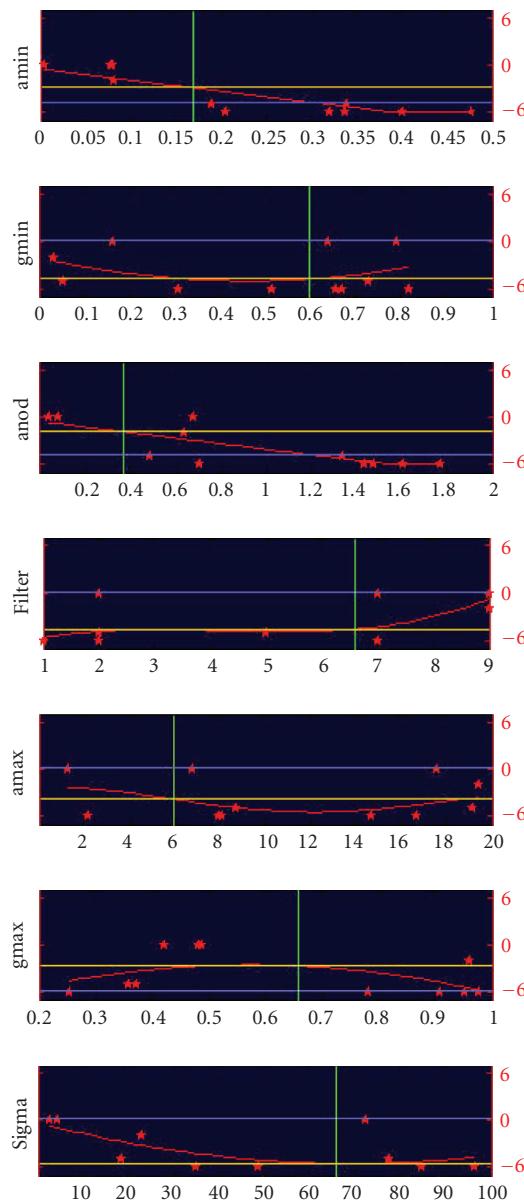


FIGURE 2.10. The 2 fitness estimation methods illustrated for a sample genotype (vertical markers): *nearest method* (horizontal grey line), *interpolation method* (horizontal light line). The y axes represent the fitness values, while the x axes represent the gene values.

A sharing algorithm has been implemented. Similarly to the sharing algorithm of the small population IEA, fitness values are weighted with a sharing factor that is

calculated from mean genotype distances within the population. Genotypes with a high mean distance to other genotypes in the population consequently have a bigger increase in their fitness. The pressure of this sharing method can be set in a configuration file, independently for each selection method. Distinct selection of individuals is also implemented and configurable.

Different versions of the genetic operators (crossover and mutation) have been implemented.

- (i) *Crossover.* “*Random*”: new individuals are a weighted combination of their parents. The weights are randomly chosen in $[0, 1]$. “*Swap*”: special case of random crossover. Parent genes are randomly swapped to generate children genotype. “*Factory*”: this method builds new genotypes out of the best genes from two parent individuals. The necessary fitness for individual genes is taken from the interpolating polynomials described earlier.
- (ii) *Mutation.* “*Random*”: Gaussian perturbation of each gene with a given σ . “*Preferred area*”: Gaussian perturbation of each gene towards its user range. There is no effect on a gene when it is already located inside the area set by the user.

2.10. Experiments

Quantitative evaluations are rather difficult to perform on interactive evolutionary algorithms. To be able to evaluate the efficiency of the fitness map scheme or, to some extent, compare the small population IEA with the large population IEA, experiments were made in a noninteractive way.

The two algorithms were run on several noisy images, for which the original “nonnoisy” images were available, and for various parameter settings.

The noninteractive software. For these tests, the software was slightly modified. User evaluations were replaced by automatic evaluations. A user fitness is therefore imitated by the calculation of a phenotypic distance between the noisy images and their corresponding original images (typically a L2 distance between images). The two presented versions of the IEA were set to run 30 generations on every noisy image and for every parameter setting. In each generation, the minimum phenodistance was collected in order to produce a convergence curve. This was repeated for at least 30 times. Afterwards a mean curve of convergence was calculated. The 2 IEAs have been compared on the basis of these average curves.

Parameters. The influence of the population size parameter has been analyzed, the parameter setting used for the tests is the following.

- (i) Large population IEA:
 - (a) population size: 16, 32, 64, and 128 Individuals;
 - (b) parent selection: rank selection (as presented in Section 2.9);

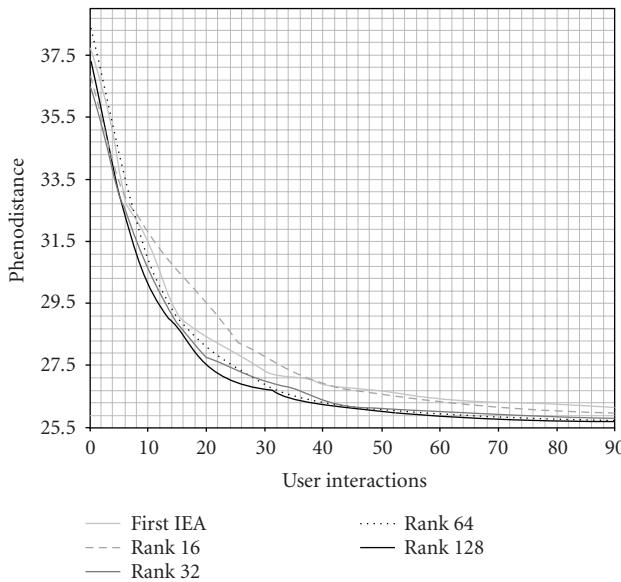


FIGURE 2.11. Comparison of mean convergence for different population sizes. Original image: Sommet 256. Noisy image: Sommet 256 with Gauss $\sigma = 20$.

- (c) offspring size: 90% of parent generation;
- (d) image selection: fittest selection;
- (c) fitness map interpolation: nearest;
- (e) use of “super individual”: in each generation the image with the lowest phenodistance to the original image is set as super individual;
- (g) *one generation is equivalent to 5 user interactions.*
- (ii) Small population IEA:
 - (a) population size: 6 individuals;
 - (b) parent selection: fittest 3;
 - (c) offspring size: 3 individuals;
 - (d) *one generation is equivalent to 3 user interactions.*

Results. To ensure a fair comparison between the two algorithms, the average curves of convergence are plotted with respect to the number of user interactions (i.e., user evaluations) instead of the generations number.

Figures 2.11, 2.12, and 2.13 show a clear improvement of the minimization behavior for the fitness map scheme, the larger the population, the more efficient.

The loss of precision of the fitness calculation based on the fitness map, which is a very rough approximation of the user, or phenotypic (for the automated version), fitness, is compensated by the exploration capabilities of a larger population.

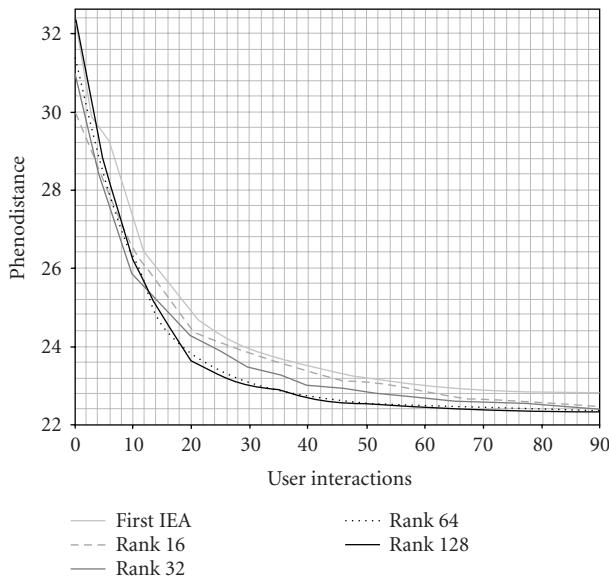


FIGURE 2.12. Comparison of mean convergence for different population sizes. Original image: Lena 256. Noisy image: Lena 256 with Gauss $\sigma = 25$.

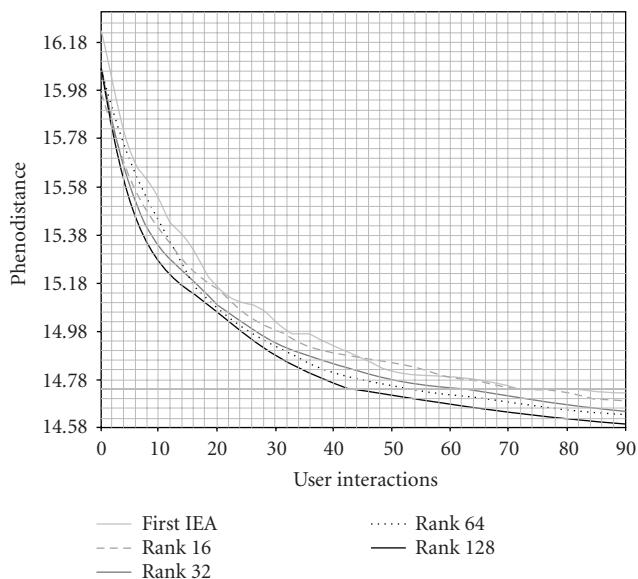


FIGURE 2.13. Comparison of mean convergence for different population sizes. Original image: Mars 256. Noisy image: Mars 256 with Gauss $\sigma = 30$.

This improved exploration capability has also been noticed in a qualitative manner by users.

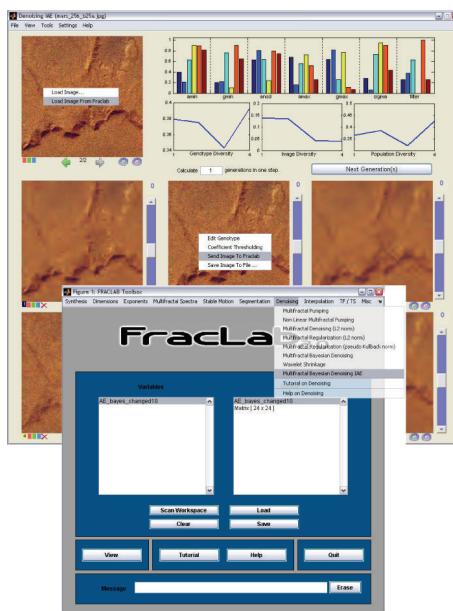


FIGURE 2.14. The presented IEA integrated with Fraclab.

2.11. Conclusion

This work about the use of evolutionary computation schemes for signal and image denoising leads us to several conclusions that may be considered from a wider point of view. First of all, it has been made evident, if necessary, that evolutionary schemes are efficient in signal and image analysis basic tasks, as far as we deal with complex optimization problems. But of course, as this choice implies heavy computational costs, such a technique may not be convenient in cases where short response time is required.

Another point, that has been raised, is the interest and efficiency of interactive schemes in image and signal processing: for subtle tasks where computational measurement cannot accurately reflect the judgment of the end-user, which is actually the case for image denoising, an IEA can be a solution. Once again, however, a careful design of the EA components and user interaction schemes is necessary. For instance, the manipulation of a much larger population in conjunction with the use of rough approximations of the user fitness provides a solution to the “user bottleneck” problem.

This work also defends a viewpoint on signal and image analysis tasks, in terms of semiautomatic procedures where an end-user is involved in order to constrain the analysis towards aims for which numerical models are not available. Such an analysis may additionally have backward consequences into noninteractive procedures. For example, the fitness map scheme can be easily generalised to other applications, including noninteractive ones where exact fitness calculation is computationally expensive.

The IEA presented in this work is freely distributed in the Fraclab toolbox, see Figure 2.14, available for download at <http://complex.inria.fr>.

Acknowledgments

The authors are grateful to Mario Pilz, Pierre Grenier, Yann Landrin-Schweitzer, and Pierrick Legrand for their contribution, technical support, and help to integrate the software into Fraclab.

Bibliography

- [1] P. J. Angeline, "Evolving fractal movies," in *Proceedings of the 1st Annual Conference on Genetic Programming (GP '96)*, J. R. Koza, D. E. Goldberg, D. B. Fogel, and R. L. Riolo, Eds., pp. 503–511, MIT Press, Cambridge, Mass, USA, 1996.
- [2] W. Banzhaf, "Interactive evolution," in *Handbook of Evolutionary Computation*, chapter 2, pp. 1–6, Oxford University Press, New York, NY, USA, 1997.
- [3] G. Brown, G. Michon, and J. Peyrière, "On the multifractal analysis of measures," *Journal of Statistical Physics*, vol. 66, no. 3-4, pp. 775–790, 1992.
- [4] J. Chapuis and E. Lutton, "ArtiE-fract: interactive evolution of fractals," in *Proceedings of the 4th International Conference on Generative Art*, Milano, Italy, December 2001.
- [5] P. Collet, E. Lutton, M. Schoenauer, and J. Louchet, "Take it EASEA," in *Proceedings of the 6th International Conference on Parallel Problem Solving from Nature (PPSN '00)*, vol. 1917, pp. 891–901, Springer, Paris, France, September 2000.
- [6] K. Daoudi, J. Lévy Véhel, and Y. Meyer, *Construction of Functions with Prescribed Local Regularity*, Constructive Approximation, Nashville, Tenn, USA, 1989.
- [7] R. A. DeVore and B. Lucier, "Fast wavelet techniques for near-optimal image processing," in *Proceedings of IEEE Military Communications Conference (MILCOM '92)*, vol. 3, pp. 1129–1135, San Diego, Calif, USA, October 1992.
- [8] D. L. Donoho, "De-noising by soft-thresholding," *IEEE Transactions on Information Theory*, vol. 41, no. 3, pp. 613–627, 1995.
- [9] L. Gagnon and F. D. Smaili, "Speckle noise reduction of airborne SAR images with symmetric Daubechies wavelets," in *Signal and Data Processing of Small Targets 1996*, vol. 2759 of *Proceedings of SPIE*, pp. 14–24, Orlando, Fla, USA, April 1996.
- [10] B. Guiheneuf and J. Lévy Véhel, "2-microlocal analysis and applications in signal processing," in *Proceedings of International Conference on Wavelet*, Tanger, Morocco, 1997.
- [11] S. Jaffard, "Pointwise smoothness two-microlocalization and wavelet coefficients," *Publicacions Matemàtiques*, vol. 35, no. 1, pp. 155–168, 1991.
- [12] Y. Jin, "A comprehensive survey of fitness approximation in evolutionary computation," *Soft Computing*, vol. 9, no. 1, pp. 3–12, 2005.
- [13] S. Kamohara, H. Takagi, and T. Takeda, "Control rule acquisition for an arm wrestling robot," in *Proceedings of the IEEE International Conference on Systems, Man and Cybernetics (SMC '97)*, vol. 5, pp. 4227–4231, Orlando, Fla, USA, October 1997.
- [14] Y. Landrin-Schweitzer, P. Collet, and E. Lutton, "Interactive GP for data retrieval in medical databases," in *Proceedings of the 6th European Conference on Genetic Programming (EuroGP '03)*, pp. 93–106, Essex, UK, April 2003.
- [15] J. Lévy-Véhel, "Fractal image encoding and analysis: a NATO ASI series book," in *Introduction to the Multifractal Analysis of Images*, Y. Fisher, Ed., chapter 17, Springer, New York, NY, USA, 1996.
- [16] J. Lévy Véhel, "Fractal approaches in signal processing," *Fractals*, vol. 3, no. 4, pp. 755–775, 1995.
- [17] J. Lévy Véhel and B. Guiheneuf, "Multifractal image denoising," in *Proceedings of the 10th Scandinavian Conference on Image Analysis (SCIA '97)*, Lappeenranta, Finland, June 1997.
- [18] J. Lévy Véhel and P. Legrand, "Bayesian multifractal signal denoising," in *Proceedings of IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP '03)*, vol. 6, pp. 177–180, Hong Kong, April 2003.

- [19] J. Lévy Véhel and P. Legrand, "Hölderian regularity-based image interpolation," in *Proceedings of IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP '06)*, vol. 3, pp. 852–855, Toulouse, France, May 2006.
- [20] J. Lévy Véhel and E. Lutton, "Evolutionary signal enhancement based on Hölder regularity analysis," in *Applications of Evolutionary Computing, EvoWorkshops: EvoCOP, EvoFlight, EvoIASP, EvoLearn, and EvoSTIM (EvoWorkshops '01)*, vol. 2037 of *Lecture Notes in Computer Science*, pp. 325–334, Como, Italy, April 2001.
- [21] P. Legrand, E. Lutton, and G. Olague, "Evolutionary denoising based on an estimation of Hölder exponents with oscillations," in *EvoWorkshops: EvoBIO, EvoCOMNET, EvoHOT, EvoIASP, EvoINTERACTION, EvoMUSART, and EvoSTOC*, vol. 3907, pp. 520–524, Budapest, Hungary, April 2006.
- [22] Y. Landrin-Schweitzer, P. Collet, E. Lutton, and T. Prost, "Introducing lateral thinking in search engines with interactive evolutionary algorithms," in *Proceedings of the Annual ACM Symposium on Applied Computing (SAC '03)*, pp. 214–219, Melbourne, Fla, USA, March 2003.
- [23] E. Lutton, P. Grenier, and J. Lévy Véhel, "An interactive EA for multifractal Bayesian denoising," in *EvoWorkshops: EvoBIO, EvoCOMNET, EvoHOT, EvoIASP, EvoMUSART, and EvoSTOC*, vol. 3449, pp. 274–283, Lausanne, Switzerland, March–April 2005.
- [24] J. J. Merelo, EO: Granada University, <http://geneura.ugr.es/merelo/EO.html>.
- [25] N. Monmarche, G. Nocent, G. Venturini, and P. Santini, "On generating HTML style sheets with an interactive genetic algorithm based on gene frequencies," in *Proceedings of European Conference on Artificial Evolution (AE '99)*, C. Fonlupt, J. K. Hao, E. Lutton, E. Ronald, and M. Schoenauer, Eds., vol. 1829 of *Lecture Notes In Computer Science*, pp. 99–110, Springer, Dunkerque, France, November 1999.
- [26] C. J. Oliver, "Information from SAR images," *Journal of Physics D*, vol. 24, no. 9, pp. 1493–1514, 1991.
- [27] R. Poli and S. Cagnoni, "Genetic programming with user-driven selection: experiments on the evolution of algorithms for image enhancement," in *Proceedings of the 2nd Annual Genetic Programming Conference (GP '97)*, pp. 269–277, Stanford, Calif, USA, July 1997.
- [28] S. Rooke, "The evolutionary art of Steven Rooke," <http://www.azstarnet.com/~srooke/>.
- [29] Y. Jamont, R. Biojout, Y. Semet, E. Lutton, and P. Collet, "Artificial ant colonies and E-learning: an optimisation of pedagogical paths," in *Human Computer Interaction International (HCII '03)*, Crete, Greece, June 2003.
- [30] K. Sims, "Interactive evolution of dynamical systems," in *Proceedings of the 1st European Conference on Artificial Life*, pp. 171–178, Paris, France, December 1991.
- [31] K. Sims, "Artificial evolution for computer graphics," *Computer Graphics*, vol. 25, no. 4, pp. 319–328, 1991.
- [32] H. Takagi, "Interactive evolutionary computation: system optimization based on human subjective evaluation," in *Proceedings of IEEE International Conference on Intelligent Engineering Systems (INES '98)*, pp. 1–6, Vienna, Austria, September 1998.
- [33] H. Takagi and M. Ohsaki, "IEC-based hearing aid fitting," in *Proceedings of the IEEE International Conference on Systems, Man and Cybernetics (SMC '99)*, vol. 3, pp. 657–662, Tokyo, Japan, October 1999.
- [34] S. J. P. Todd and W. Latham, *Evolutionary Art and Computers*, Academic Press, Amsterdam, The Netherlands, 1992.
- [35] M. Wall, GAlib: MIT, <http://lancet.mit.edu/ga/>.

Evelyne Lutton: APIS team, INRIA Futurs, Parc Orsay Università, 4 rue Jacques Monod, 91893 Orsay Cedex, France

Email: evelyne.lutton@inria.fr

Jacques Lévy Véhel: APIS team, INRIA Futurs, Parc Orsay Università, 4 rue Jacques Monod, 91893 Orsay Cedex, France

Email: jaques.levy_vehel@inria.fr

3

Sub-machine-code genetic programming for binary image analysis

Stefano Cagnoni, Monica Mordonini, and Giovanni Adorni

3.1. Introduction

Applications which require real-time or quasi-real-time processing are more and more frequent in computer vision, as well as in many other application fields, thanks to the continuous increase of computing power available to programmers. However, at the same time, such an increase keeps being challenged by the quantity of data on which applications must be run, which is increasing with a similar, when not faster, trend. An outstanding example of such a situation is offered by the recent advances in image sensors, which has led to an increase in resolution available for digital pictures close to an order of magnitude in the last four/five years.

Therefore, even when algorithms of polynomial (or lower) complexity are used, the need for efficient computing architectures, on the hardware side, or computing paradigms, on the software side, is still a critical problem in most applications.

When the data to be processed or generated are either synthetic images (as in computer graphics) or real-world images (as in image processing and computer vision), SIMD (Single Instruction, Multiple Data) architectures, in which the same instruction is executed in parallel on a large array of data, are often used to dramatically speed up processing time. The ever-growing availability of multimedia applications, even to that huge range of users which typically use low-end PCs, has led the main microprocessor manufacturers to introduce a specific set of instructions, which actually implement the SIMD paradigm, in their mainstream products. This is the case, for example, of Intel and AMD, who have added specific multimedia extensions (MMX and 3DNow!, resp.) to the instruction sets of their PC processors since 1997. Another relevant SIMD architecture which is widely used is the cellular automaton [12, 14, 15], which has been widely used in several applications, including image processing and analysis, in both software [1, 4, 13] and hardware implementations (e.g., the CAM-8 cellular automata machine [8]).

A print edition of this book can be purchased at

<http://www.hindawi.com/spc.8.html>

<http://www.amazon.com/dp/9774540018>

In several software implementations of the SIMD paradigm, long words are used to pack binary data, on which bitwise Boolean or other bit-manipulating instructions from the native instruction set of a processor are applied. In the following, we focus on Sub-machine-code genetic programming (SmcGP) [10, 11], a variant of genetic programming, which implements the SIMD computation paradigm. In particular, we describe a general framework within which binary classifiers or sets of binary classifiers can be evolved, which can be applied to the solution of a large number of problems. One of the main features of SmcGP is the capability of evaluating several (nonindependent) individuals in parallel, which allows such an approach to explore the search space very effectively. An even more relevant property of the approach under consideration is the capability to produce highly efficient accurate classifiers, due to the use of a function set which allows for either a direct or a very efficient translation into machine code of the evolved programs.

In this chapter, we illustrate the potentials and the limitations of the approach by describing results obtained in designing automatically a set of binary classifiers for low-resolution characters and an image-preprocessing procedure, also based on the use of a binary classifier. Both tasks have been designed to be possible parts of a license-plate recognition system. Results obtained in such tasks are compared with those obtained by a “human-designed” plate-recognition system we developed in our laboratory [2].

3.1.1. Sub-machine-code genetic programming

Given a problem to be solved that can be mapped onto an optimization problem, evolutionary computation (EC) techniques use different encoding schemes (genotypes) to allow for different spaces of solutions (phenotypes) to be explored. While, for example, genetic algorithms (GAs) typically optimize the parameters of a function whose structure has been formerly defined, in genetic programming (GP) [3, 7], the phenotype of each individual in the population is a full program. Several GP paradigms have been described, the most commonly used is the one originally proposed by Koza [7], where programs are encoded as syntactic trees or, equivalently, as prefix-notation LISP-like functions.

Genetic programming paradigms are usually computationally very intensive. The higher computation load imposed by the optimization phase with respect, for example, to GAs is, firstly, due to the much wider search space it spans. Secondly, tree-like encoding of solutions makes decoding, during fitness evaluation, as well as crossover and mutation operators, much less efficient and CPU-demanding with respect to binary-string encoding used in GAs. Therefore, there is great interest, on the one hand, in developing GP implementations which improve the computational efficiency of evolution.

On the other hand, the increasing need for high-performance programs that perform real-time tasks has fostered the development of GP variants in which data encoding and representation are such that programs, besides being optimized with respect to a given fitness (cost) function, can also be executed very efficiently at runtime. In view of this, approaches to GP that imply some degree of parallelism,

both in executing the evolutionary algorithm and in the structure of the resulting programs, have been proposed.

SmcGP aims at exploiting the intrinsic parallelism of bitwise instructions of sequential CPUs and can be run effectively on traditional computer architectures. Inside a sequential N -bit CPU, for example, each bitwise operation on integers is performed by concurrently activating N logic gates of the same kind. Because of this, the application of a sequence of bitwise logical operators to a N -bit integer is equivalent to executing the same program on N 1-bit operands in parallel. In practice, SmcGP is a form of GP in which functions based on bitwise operators applied to packed representations of 1-bit data are evolved using that EC paradigm. Therefore, SmcGP can produce programs that are intrinsically parallel and based on operators that have a direct machine code translation. That makes such programs computationally very efficient.

SmcGP efficiency is further increased by the closure requirement which constrains GP. Such a constraint requires that any GP-evolved function that operates on a certain input data type also produce an output of the same type. Therefore, in using GP to optimize a binary function $f : \{0, 1\}^N \rightarrow \{0, 1\}$, each phenotype actually represents a set of N (nonindependent) functions for which fitness can be independently evaluated.

Because of these properties, SmcGP can be used effectively in applications in which the same operations must be performed on blocks of binary data that can be packed, for instance, into a long integer variable. This is the case for binary pattern recognition or binary image processing problems, in which 2D patterns can be processed linewise or blockwise.

3.2. Evolving binary classifiers using Sub-machine-code genetic programming

SmcGP can be used to efficiently develop high-performance binary classifiers, in terms of both accuracy and computation speed. This is a result of more general interest, since any N -class classifier of arbitrary complexity can be implemented as an ensemble of distinct specialized binary classifiers which, organized in different possible architectures with different degrees of redundancy, can perform the original, more complex, classification task.

However, the choice to use a multiclassifier approach must be corroborated by methods that produce fast, accurate classifiers very efficiently for a multiplicity of reasons. Firstly, since the final accuracy of an ensemble of classifiers depends, at least linearly, on the error rates of the single classifiers, developing ensembles of classifiers require that each component be very accurate. Secondly, even if an ensemble of classifiers usually yields better results than the corresponding single classifier, thanks to a richer and/or redundant processing of information, developing an ensemble of binary classifiers is usually much more time-consuming than developing a single, equivalent classifier. Finally, running an ensemble of classifiers is generally very computationally demanding.

We describe a general framework within which the evolutionary approach can be used to design binary classifiers which meet the above-mentioned requirements. The performance of binary classifiers thus obtained has been assessed on a low-resolution digit recognition problem and on an image “segmentation-by-classification” task, in which classifiers have not only been used as stand-alone modules, but also as building blocks for multiple-classifier architectures.

Following a typical detection scheme, each classifier is associated to one class and is required to have 1 as output when the input pattern belongs to the corresponding class, and 0 otherwise. To solve N -class problems, a set of N binary classifiers can be used; the final classification can be derived from the analysis of the response of all classifiers. This is, for instance, the typical architecture and training strategy used in N -class classifiers based on feed-forward neural networks. In the approach described in this chapter, however, classifiers are evolved independently of one another, differently from neural networks in which paths leading from the input to the output layer share several weights, allowing classifiers to be trained concurrently.

The ideal situation for such a multiclassifier architecture occurs when only the classifier corresponding to the class of the input pattern outputs 1, while all others give 0 as output. As pointed out in [5], where this kind of classification architecture is described in detail with reference to the use of classifiers evolved by GP, there are, quite intuitively, two trivial cases in which this strategy fails. One occurs when no classifier produces a high output, while the other occurs when more than one classifier produce 1 as output. The problem of deriving a final decision in this case can be tackled by several different approaches, such as a hierarchy of increasingly specialized classifiers or an “*a posteriori*” statistical approach.

Such architectures usually perform classification based on criteria similar to those used in sports tournaments. If one looks upon the output of the classifier as the result of a match between the two classes under consideration, the decision could be based, for example, upon a knock-out or a round-robin tournament mechanism. A more computationally expensive but usually more performing approach is to record all outputs of the classifier set and combine them into a pattern that is then classified by a so-called “stacked generalizer” [16].

Therefore, from the point of view of classification strategy, the general architecture of the evolutionary classifiers considered in this chapter is quite conventional. The peculiar features of the approach are related with the high degree of parallelism in the computation performed by each individual and with the criterion by which the output of each individual, and therefore its fitness function, is computed. Regarding the former property, SmcGP performs parallel computation by applying bitwise operators to packed representations of arrays of 1-bit data. More precisely, if P is the dimension in bits of the input space in SmcGP, the function computed by each individual I_k is a function $f_k : \{0, 1\}^P \rightarrow \{0, 1\}^S$ ($S > 1$). Each f_k can be seen as a set of S alternative binary-output functions $f_{ki} : \{0, 1\}^P \rightarrow \{0, 1\}$ computed concurrently at each function evaluation. The output space size S is equal to the size of the word into which 1-bit data are packed.

Therefore, decoding each individual I_k implies two steps. In the first one, the function encoded by the corresponding representation is decoded. In the second one, each bit of the result obtained in the first step is considered, in turn, as the output of the classifier, and the corresponding binary-output function is taken into consideration. As long as a fitness function F is defined for the problem at hand, a different fitness value $F(f_{ki})$ can therefore be associated to each solution. The fitness $F(I_k)$ of each individual is equal to the maximum fitness obtained in the second step,

$$F(I_k) = \max_i (F(f_{ki})). \quad (3.1)$$

This means that while only one fitness value is assigned to each individual I_k , S fitness function evaluations are performed for each evaluation of I_k . This implies that defining fitness as in (3.1) can be interpreted as choosing one of the bits of the output pattern as the actual 1-bit output of an individual.

Besides being beneficial from the point of view of computation efficiency, this strategy favors extraction of the most relevant features of the patterns under classification. This can be explained considering that, in SmcGP, there exists a direct morphologic and geometric correspondence between input and output. Even when SmcGP spans a larger function space than the one defined by the encoding of inputs into one long word, and evolves complex functions that are made up of building blocks that operate on different “slices” of the input pattern and keeps such a correspondence for each of the words into which the whole input is divided, operations performed on those slices are local.

Because of these properties, the fact that one bit of the output pattern provides the best fitness generally implies that the area of the input pattern by which the value of such a bit is most influenced contains the most significant feature for the classifier under consideration.

3.3. Applications of SmcGP to license-plate recognition

We have used binary classifiers evolved by SmcGP to build or integrate modules which solve two of the main tasks required by a license-plate segmentation system: license-plate detection and license-plate character recognition. As a comparison for the results obtained by the classifiers evolved by SmcGP, we have used the corresponding modules of the Apache plate-recognition system which we had formerly developed [2]. The structure of Apache is reported in Figure 3.1. The top part of the graph corresponds to image preprocessing and license-plate detection, while the bottom part corresponds to the subsequent phases of character segmentation and classification.

At first glance, character classification for license plate-recognition [9] seems to have much in common with traditional optical character recognition (OCR) applications. However, differently from OCR, in which character classification occurs after a text has been scanned at high resolution, only very low-resolution patterns are available for classification in license-plate recognition, usually obtained from

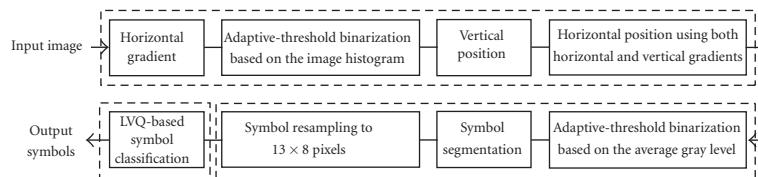


FIGURE 3.1. Structure of the Apache plate recognition system.



FIGURE 3.2. Examples of the original (top) and binarized (bottom) patterns.

low-quality snapshots, altered by optical distortions and perspective effects (see Figure 3.2 for some examples). Therefore, classification of license-plate characters is usually a much more critical task with respect to classification of characters contained in printed documents.

In studying and comparing the application of SmcGP [4] to low-resolution license-plate character classification, performance was evaluated on a large set of binary patterns of size 13×8 pixels. For practical reasons, the comparison was limited to patterns representing digits from 0 to 9, taken from a larger database collected at highway toll booths.

In a further experiment to evaluate the performances of SmcGP in designing classifiers oriented to image-analysis applications, a preprocessing stage was added to the plate-detection module of the Apache license plate recognition system. The aim of such a module is to locate the region where a license plate is most likely to be found within an image. Such a detection derives from the basic consideration that a license plate is characterized by the high density of peaks of the horizontal component of the image gradient which can be observed within it (see Figure 3.3). Therefore, after removing all pixels belonging to the background by performing an image difference which lets only objects which are moving within the scene be visible, the region occupied by the license plate, if any, is commonly the rectangular region, shaped compatibly with the width-to-height ratio of a license plate, where the density of such peaks is highest. In practice, a binarized gradient image is generated by thresholding the horizontal-gradient image, then rowwise and columnwise statistics of the distributions of the pixels which are set to one are computed to isolate the license-plate region (see the following and [2] for more details).

Pixels having significant gradient values are usually quite few and sparse in images like the ones in the picture, which result from a background-removing image difference. This increases the chances of making false detections, since just a few “noisy” pixels may be enough to turn the algorithm attention to a region different from the one to be detected. The incidence of false detections can be reduced



FIGURE 3.3. A typical input image of the Apache system and the corresponding horizontal-gradient image.

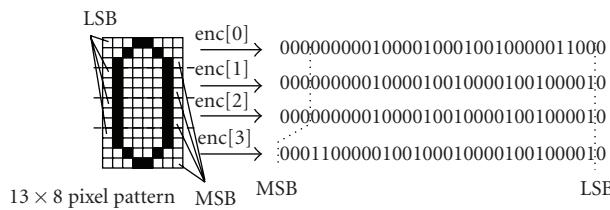


FIGURE 3.4. Encoding of a character as four 32-bit integers.

if we make the statistics about the distribution of significant pixels more robust by increasing their density. This can be obtained by training a binary classifier to output one for each pixel belonging to the plate region and zero elsewhere.

Doing so, if the training results are reasonably good (very high sensitivity and specificity are not necessary, even if obviously preferable), the chances of having a lower density of pixels set to 1 in the plate region to be detected than in some other region geometrically compatible with the plate shape are much fewer.

For both license-plate character recognition and license-plate detection, a binary classifier or a set of binary classifiers evolved by SmcGP have been used to solve the problem. In the remainder of the section, we offer details about the implementation of the two applications, while in the subsequent one we describe and discuss the results we have obtained.

3.3.1. License-plate character recognition

To apply SmcGP to character classification, each pattern of size 13×8 pixels was encoded as four 32-bit integers. Since the total number of bits in each pattern is not a multiple of 32, the first nine rows of the pattern were encoded as the 24 least significant bits of the first three integers, while the whole fourth integer was used to encode the last four rows of the pattern, as shown in Figure 3.4. Preserving the same pixel order in the 32-bit components of the code was the only specification

TABLE 3.1. The function set (above) and terminal set (below) used to evolve the classifiers with SmcGP.

Functional	Arity	Notes
AND	2	Bitwise AND
OR	2	Bitwise OR
NOT	1	Bitwise NOT
XOR	2	Bitwise XOR
SHxn	1	$x \in \{L, R\}, n \in \{1, 2, 4\}$

Terminal	Type	Notes
pat[0]	Term	Rows 1–3
pat[1]	Term	Rows 4–6
pat[2]	Term	Rows 7–9
pat[3]	Term	Rows 10–13
R1	ERC	Unsigned long
1	Constant	32-bit constant 1
0	Constant	32-bit constant 0

for the encoding. The encoding would have been equally effective if, for example, pixels had been packed left to right, or columnwise.

The closure requirement of GP [3] imposes that the output of each classifier, that operates on unsigned long words, be an unsigned long as well. Therefore, using 32-bit encoding, for each evaluation of an individual I_k , 32 binary-output functions f_{ki} are actually computed.

The function set was composed by the main bitwise Boolean operators and by a set of circular shift operators SHxn, in which the 32-bit word LSB is considered to be adjacent to the MSB, characterized by different shift direction ($x = L$ (left) or R (right)) and entity ($n = 1, 2$, or 4 bits). The terminal set (see Table 3.1) was composed by the 4 unsigned long integers into which the input pattern had been encoded, an unsigned long integer ephemeral random constant (ERC) [3, 7] which can take values within the whole range of 32-bit unsigned integers, and the 32-bit constants 1 and 0.

The fitness function for function f_{ki} was defined as

$$F(f_{ki}) = 1 - \sqrt{\frac{FP_{ki}^2 + FN_{ki}^2}{N_p^2 + N_n^2}}, \quad (3.2)$$

where FP_{ki} is the number of false positives generated by f_{ki} in classifying the training set, FN_{ki} the number of false negatives, N_p the number of positive examples, and N_n the number of negative examples in the training set. This choice of the fitness function was made after comparing it to several other possible functions based on the same data, and was found to produce very specific classifiers, with very high positive predictivity, and a very limited number of ambiguous cases in which more than one classifier produce a high output. This can be easily explained

by considering that, with a roughly uniform distribution of the patterns in the training set, negative cases are about 9 times as many as positive cases. Therefore, since the numerator of the fraction tends to minimize errors and privileges situations in which the number of false positives and of false negatives are similar, specificity, which is proportional to the number of false negatives, is greatly favored.

A population was evolved, made up of 1000 individuals, randomly initialized with constraints on minimum (3) and maximum (7) tree depth. A crossover rate of 80%, a mutation rate of 2%, and a reproduction rate of 18% were used during evolution. The strategy chosen for selection was tournament selection with tournament size equal to 7. Each classifier was evolved for 1000 generations. At least two runs for each classifier were performed.

The classifiers were evolved using *lil-gp1.01* [17], a popular package that implements Koza-like (i.e., LISP-like or tree-like) genetic programming. Depending on the resulting classifier length, and therefore on the complexity of the classification task to be performed, each run of the GP took from 10 to 18 hours on a 600 MHz Pentium-III PC.

At the end of evolution, the best classifier for each class was converted from prefix notation, typical of syntactic trees, to infix notation, and translated into C language functions to allow for its compilation.

3.3.2. License-plate detection

As reported above, the goal of the application is to evolve a classifier which implements an additional preprocessing stage, refining results obtained by computing the horizontal gradient, whose ideal output is an image in which all pixels belonging to the license plate are set to 1, while all others are set to zero. This is a typical problem of region-based segmentation by classification, in which significant regions of the image under consideration are labeled as a result of pixel-level classification, and possibly of some trivial post-processing which removes “noise,” typically represented by isolated pixels classified differently from the ones which surround them.

Ideally, neglecting perspective effects due to camera geometry, one should expect to obtain as output an image showing a black rectangle on white background (or vice versa), having the same size ratio and location as the license plate to be detected in the image.

Formally, we could state the problem as evolving a function $S(B_h(x, y))$ of the binarized gradient image $B_h(x, y)$ such that

$$S(B_h(x, y)) = \begin{cases} 1 & \text{if } (x, y) \text{ belongs to the license plate,} \\ 0 & \text{otherwise.} \end{cases} \quad (3.3)$$

The classifier evolved to perform such a task has the same structure as one of the classifiers used to recognize digits which have been described in the previous

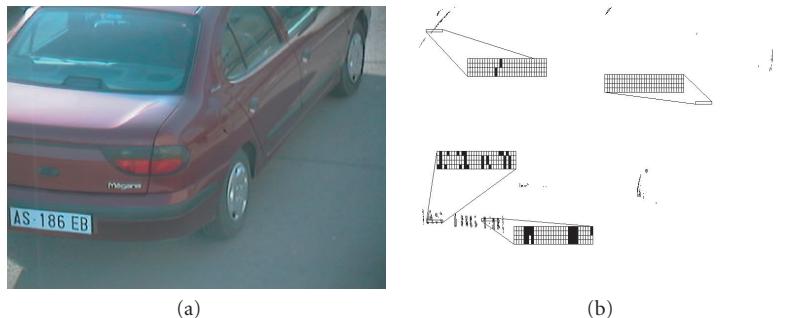


FIGURE 3.5. Encoding of image regions as 4 32-bit integers.

section. The terminal set is also the same, while the function set has been extended by introducing also the NAND and NOR functions, as well as the 32-bit logical inversion function $N32$, having a single argument, which returns 1, if all bits of the argument are set to 0, and 0 otherwise.

The encoding of the image windows in the four terminals which directly derive from input data is also slightly different. As shown in Figure 3.5, the four *unsigned long int* terminals now encode a rectangular region of size 32×4 , defining a neighborhood of the pixel which is located on the upper left corner of the region. This means that the value of the pixel of coordinates (x, y) in the output image, $O(x, y)$, is computed based on the values of pixels belonging to the rectangular window B_w of B_h whose upper left and lower right pixels have coordinates (x, y) and $(x + 31, y + 3)$, respectively.

A set of 130 images of rear views of car (as the one shown in Figure 3.5) were considered. Of these, 80 were used to generate the training set. For each of the 80 training images, 100 windows were extracted, encoded as described above, and used as examples. Of these, 60 were taken from the license-plate region, 30 from an extended region surrounding the license plate (where the chance of finding high-gradient pixels not belonging to the plate is highest), and 10 from anywhere in the image. Among the windows which were extracted, “empty” windows with no pixel set to 1 were purged, since they carry no relevant information and are intrinsically ambiguous, being found both within and outside the license-plate region. In the end, the training set was made up of 5190 “nonempty” encoded windows, of which 366 were negative and 4824 were positive examples. The results were evaluated on all pixels of the images in the test set.

The parameters regulating evolution were virtually the same as in the previous experiment (1000 individuals, 80% crossover, 17% clonation, 3% mutation). The fitness function was also very similar, except for a parsimony term which penalizes larger trees:

$$F = 1 - \sqrt{\frac{FP^2/N_n^2 + FN^2/N_p^2}{2}} - \frac{\text{Tree_Size}}{10000000}. \quad (3.4)$$

TABLE 3.2. Performances of the SmcGP-based classifiers. Processing time (in μ s) is computed as average runtime in 10 executions of the classifier on the test set, as recorded by *gprof* on a PIII 600 MHz PC.

	0	1	2	3	4	5	6	7	8	9
Specificity	99.87	99.49	99.76	99.65	99.62	99.45	99.56	99.87	99.53	99.73
Sensitivity	96.01	98.00	95.21	96.41	95.81	96.21	95.41	94.81	91.82	96.21
Tree nodes	45	115	290	154	71	195	232	96	304	210
Proc. time	1.40	1.80	2.59	1.60	1.60	3.79	3.99	1.20	3.79	3.59

3.4. Results

The performance of the corresponding modules of the Apache system was considered as a reference for evaluating the results obtained applying SmcGP to evolve binary classifiers in the two applications under consideration. The set of binary classifiers evolved for character recognition were directly compared to the corresponding Learning Vector Quantization (LVQ) [6] neural network which actually performs the same task in the Apache system. As concerns the license-plate detection, the comparison was slightly different, since the classifier evolved using SmcGP acts as a preprocessing module which is added after the one which computes the horizontal gradient and right before the actual license-plate locator. Therefore, in the absence of a direct correspondence with a module of the Apache system, the results were evaluated quantitatively in terms of the increase of density of detected pixels within the license plate region, taking into consideration also the inevitable corresponding increases of false detections and of computation time, with respect to the horizontal-gradient image.

3.4.1. License-plate character recognition

The set of 10 classifiers that were used in the tests were made up by the best classifiers that could be obtained in the two or more 1000-generation runs of the algorithm performed for each class. Under this point of view, it should be noticed that, despite allowing 1000 generations per run, in most cases the best classifier emerged within the first 250 generations.

The tree-encoded classifiers evolved using SmcGP were tested after being translated in C code and then compiled. Table 3.2 reports the specificity and sensitivity of the ten classifiers, along with data related with their complexity and computation efficiency. In particular, the table reports the number of nodes in the tree-like representation produced by *lil-gp*, and the processing time in microseconds required by each classifier on a Pentium III-600 MHz PC running Linux kernel 2.2 and using the *gcc 2.95* C compiler.

Table 3.3 reports the performance, on the test set, of the reference LVQ classifier in the same form, with data derived from the global confusion matrix of the classifier.

TABLE 3.3. Performance of the reference LVQ classifier on the test set.

	0	1	2	3	4	5	6	7	8	9
Specificity	99.80	99.97	99.93	99.91	99.87	99.82	99.71	99.76	99.82	99.93
Sensitivity	99.00	99.00	99.00	98.00	99.00	98.20	99.00	99.40	98.80	97.41

As can be noticed, specificity of SmcGP classifiers is always very high (above 99.4%). Processing time varies from 1.2 to 3.99 microseconds and is not exactly proportional to the classifier size, since basic bitwise Boolean functions can be virtually run in one clock tick, while circular shift operators are at least three times as demanding, since they are composed by two basic shifts and one OR operation.

The basic binary classifier set could reach a nonambiguous output configuration (only one classifier produced 1 as output), for 95.31% of patterns in the test set, with a classification accuracy of 98.68%. Since results in the basic configuration were already quite good for the patterns that could be directly classified, the classifications that could be obtained directly were considered as final. Since the focus of our research was mainly on evolutionary development of efficient binary classifiers, in the ambiguous cases, we used the LVQ reference classifier as “tie-breaker,” without considering more complex classifier architectures. Anyway, this was necessary only in a very limited number of cases (4.69%), which helped keep computational efficiency high, since the LVQ classifier is much more computationally demanding than the SmcGP-based one.

After application of this two-stage classification strategy, the global accuracy on the whole test set was 98.30%, versus 98.68% achieved by the LVQ classifier. The processing time for the whole test set was 0.15 second on a Pentium III 600 MHz (computed averaging over 10 runs of the classifier), which is about 10 times faster than the LVQ reference classifier, that requires 1.37 seconds to perform the same task on the same PC. It should be noticed that both processing times include time needed to read data from a file, which is actually smaller (4 32-bit integers to be read per pattern) in the case of the SmcGP-based classifiers than in the case of the LVQ classifier (104 8-bit short integers to be read per pattern). However, in the former case, the time required to convert the 5% of patterns that cannot be classified in the first stage from the 4-long integer representation to the 104-char representation needed by the LVQ classifier is also included.

3.4.2. License-plate detection

Seven runs of SmcGP were performed with the parameters set as reported in Section 3.3.2, after which the best program which had been evolved was evaluated. The performances of such a program on the training set are reported in Table 3.4, in terms of sensitivity (the percentage of detections with respect to the number of pixels belonging to the plate), specificity (the percentage of negative responses with respect to the number of pixels not belonging to the plate), fitness, and program size.

TABLE 3.4. Performance of the best individual on the training set.

Sensitivity	83.85%
Specificity	88.25%
Fitness	0.142296
Program size (nodes)	1087

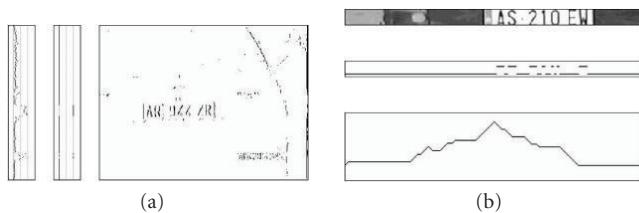


FIGURE 3.6. The two steps of the plate detection algorithm: detection of the vertical position (a) and of the horizontal position (b).

As concerns sensitivity and specificity, results are quite good both on an absolute scale, but even more if one considers the way in which the license-plate detection algorithm works.

In fact, as shown in Figure 3.6, the plate is detected in two steps. First, the horizontal gradient image is computed and binarized, to obtain the binary image B_h . Such an image is then used to find the vertical position of the plate within the original image. To do so, the histogram representing the rowwise distribution of the density of pixels set to 1 in B_h is computed. Such a histogram is then binarized. The largest interval in which all elements of the binarized histogram are set to 1 corresponds to the horizontal band, within the captured image, where the plate is most likely to be found. Once the vertical position has been found, the segmentation algorithm scans the detected band in the original image to locate the plate also horizontally. To do so, it looks for the coordinate of the left border of the box containing the plate, making a convolution between the identified band I_b and a box B having the size which the license plate is expected to have, as a function of the height of I_b . The location where the convolution function reaches its maximum corresponds to the position within the image of the left side of the license plate.

Considering that sensitivity represents the density of pixels set to one in the license-plate region and that (1-specificity) represents the density of pixels set to one in the much vaster region outside the license plate, the performance obtained makes plate detection quite robust. It should also be noticed that, having removed “empty” regions from the training set, of which most of the background is made up, the specificity value is actually largely underestimated. In fact, a much lower sensitivity is enough to reach high detection performances, when actual specificity values are well above 95%, as they actually turn out to be when actual images are processed.

TABLE 3.5. Average number (and percentage) of true positives (above) and of false positives (below) on the test set. The data in the third column reports the number of actual positive (above) and negative cases (below).

	Gradient (%)	SmcGP (%)	Number of pixels
License plate	367 (6.48%)	3513 (62.05%)	5661
Background	470 (0.12%)	8157 (2.00%)	407758

This is demonstrated also by Table 3.5, in which results collected by averaging the results over the whole image set are reported, for the 103 images out of 130 where B_h was not empty, that is, in which failure in detecting the license plate does not depend on the performance of the SmcGP-based module, being only due to missing input data.

In the table, it is possible to notice how much higher specificity actually is, when all pixels of the input image are considered. Of course, the fact that “empty” pixels are present also in the license-plate region and are classified as “background,”¹ justifies the corresponding worsening of sensitivity with respect to the results obtained on the training set. Furthermore, it is possible to appreciate the increase (by one order of magnitude) in the number of pixels set to 1 in the license-plate region, with respect to those which could be detected in the image B_h . Even if the number of false detections increases even more, its value is kept below 2% (specificity is therefore still 98%), which is still far from being able to significantly affect the detection results.

Even if the value of these remarks are limited just to the rather small image set on which the approach was tested, there were no problems in correctly detecting the license plate when more than 45% of the pixels in the license-plate are set to 1, which happened in all but two of the 103 images under consideration.

Figure 3.7 shows how typical output images look like. Of the two examples which are reported, one exhibits a quasiperfect behavior, extracting virtually only pixels belonging to the license-plate region. The other one, instead, is an interesting example of how even very noisy outputs can do little harm to the accuracy of the final detection. In such an image, the plate region is very well highlighted, but there are also several false detections in the background, due to the high-contrast areas in the right part of the original image produced by sunlight reflections on the car body. However, it is rather clear that the sparseness and, especially, the shape of the false-detection areas makes it virtually impossible for the detection algorithm to misclassify any such areas as the license-plate region. The license-plate detection algorithm could be made even more robust by using information about both horizontal and vertical distribution of gradient peaks concurrently instead of sequentially. However, at least with the images considered during the test of the application, such an upgrade, which would also have a nontrivial cost in terms of computation load, seems not to be necessary.

¹Along with the fact that the results are computed on a data set of which the training set is just a very limited part.

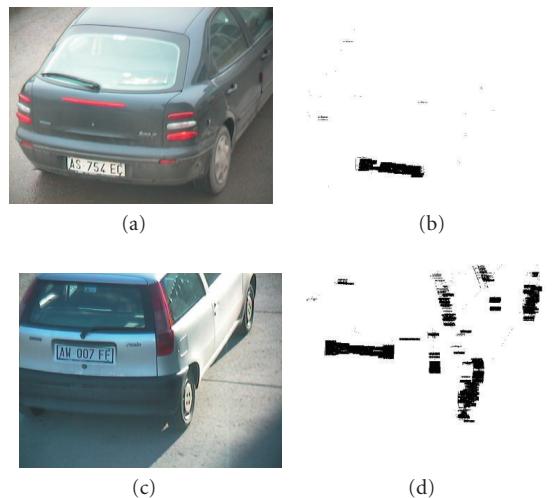


FIGURE 3.7. Typical results: the case reported above is quasi-ideal. In cases like the one reported below, despite the very “noisy” output, the distribution of the false detections is such that the license-plate region can be easily detected anyway.

As concerns computation time, the best program evolved, as can be observed by comparing its size with the size of the classifiers evolved for character classification, is a rather large function. As could be expected, one execution of a C-function directly derived from the GP tree, with no optimization, requires, on a Pentium IV 2.8 GHz PC, about as much time (2.55 microseconds) as one average execution of the character classifiers required on a Pentium III 600 MHz PC. This means that, using what, at present, is a standard PC configuration, introducing the SmcGP-based image preprocessing function may add a significant amount processing time to the computation requirements of the license-plate detection module, when a whole input image of size 768×576 pixels, as the ones typically used in the Apache system, is to be analyzed.

However, with compiler optimization on, only 0.65 second are needed to generate the output image, including loading the input image, computing the gradient image and saving the output image into a file. Furthermore, the statistics on computation time (as happened in the character classification task) have been made after implementing the function exactly as was evolved by GP. Looking more accurately at the source code, even without trying to detect possible branches of the tree which do not contribute at all to the result (which in GP terminology are usually referred to as “introns,” and whose effects on evolution, if any, are still controversial [3]), one can observe that most shift operations are built by chaining the basic shift operations which are present in the GP function set. Considering that more than half of the nodes in the tree are shift functions, and that circular shifts require more than three times as many machine cycles than the other bitwise operation, a reduction of the computation load by more than 50%, or about as much,

can be expected after simplifying the function (which is most probably what the optimizing compiler does!).

3.5. Conclusions

Even when applications have to deal with complex input data like color images, very often, at some processing stage, binary classifiers producing one binary output out of a packed array of binary data need to be developed. In computer vision and, more generally, in pattern recognition applications, such as tracking, surveillance, industrial inspection, and quality control, detection of specific objects of interest (targets, defects, etc.) is often performed by examining arrays of relatively simpler data. This happens, in general, when data are preprocessed and the results are packed into small-size binary words, when not into single bits, which are used as labels to synthesize information, as, for example, in those frequent cases in which objects can be classified just based on their shape or even on a skeletonized representation. Binary classifiers which implement a mapping $f : \{0, 1\}^N \rightarrow \{0, 1\}$ are therefore most common and of great importance in such applications.

In this chapter, we have described how SmcGP can be used to produce efficient and accurate binary classifiers, with an approach which can be easily adapted to a large number of applications. In the domains of pattern recognition and image processing, we have presented two applications to different subtasks of license-plate recognition, character recognition, and image preprocessing for license-plate detection, which, although far from being completely developed and refined, offer interesting insights about the usability and effectiveness of the approach in the practice, on hard real-world problems.

Applying SmcGP to two-dimensional binary pattern classification yielded very good results, under both points of view, even relying on a quite naive classification scheme (one set of binary classifiers). Such a scheme could be further improved either by introducing a new layer of pairwise classifiers, or by using some kind of tournament-like strategies, possibly along with a stacked generalizer, to produce the final classification. However, the results show that, even in this simple configuration, the SmcGP-based classifiers performance is very close to the performance yielded by the reference LVQ classifier, that we had previously shown to outperform other kind of classifiers (e.g., classifiers based on multilayer perceptrons trained with the back-propagation algorithm) in this particular task. As regards computation efficiency, the implicit parallelism of the single classifiers, jointly with the simplicity of the global architecture, makes the SmcGP-based classifier almost 10 times faster than the LVQ classifier.

Using SmcGP to increase robustness of a plate detection algorithm by designing an image preprocessing stage was also successful, even if the measure of the effectiveness of the approach strongly depends on the particular constraints and specifications imposed by the application. Therefore, it is difficult to evaluate the actual impact of the pros and cons of the approach, which have emerged in a specific subtask, without testing the performance of the full system integrating the modified module on an extended set of cases.

A final peculiarity of the approach lies in the natural translation into hardware which is allowed by the functions evolved by SmcGP, due to both the simplicity of the functional representation and to the availability of the basic bitwise Boolean functions as part of the instruction set of virtually all digital hardware architectures, from the very simplest microprocessors to the most powerful ones. This can be particularly important in developing embedded systems, an industrial field which is gaining greater and greater importance, as well as, more directly, in the field of evolutionary hardware.

Acknowledgments

This project has been partially funded by ASI (Italian Space Agency) under the “Hybrid Vision System for Long Range Rovering” grant, and by the Italian MIUR (Ministry of Education, University and Research) under the “Evolutionary Computation Methods for Pattern Recognition and Classification” FIL 2001 grant.

Bibliography

- [1] G. Adorni, F. Bergenti, and S. Cagnoni, “A cellular-programming approach to pattern classification,” in *Proceedings of the 1st European Workshop on Genetic Programming (EuroGP '98)*, vol. 1391 of *Lecture Notes In Computer Science*, pp. 142–150, Springer, Paris, France, April 1998.
- [2] G. Adorni, F. Bergenti, S. Cagnoni, and M. Mordonini, “License-plate recognition for restricted-access area control systems,” in *Multimedia Video-Based Surveillance Systems: Requirements, Issues and Solutions*, G. L. Foresti, P. Mähönen, and C. S. Regazzoni, Eds., pp. 260–271, Kluwer Academic Publishers, Dordrecht, The Netherlands, 2000.
- [3] W. Banzhaf, F. Francone, J. Keller, and P. Nordin, *Genetic Programming: An Introduction*, Morgan Kaufmann, San Francisco, Calif, USA, 1998.
- [4] S. Cagnoni, F. Bergenti, M. Mordonini, and G. Adorni, “Evolving binary classifiers through parallel computation of multiple fitness cases,” *IEEE Transactions on Systems, Man, and Cybernetics, Part B*, vol. 35, no. 3, pp. 548–555, 2005.
- [5] J. K. Kishore, L. M. Patnaik, V. Mani, and V. K. Agrawal, “Application of genetic programming for multiclass pattern classification,” *IEEE Transactions on Evolutionary Computation*, vol. 4, no. 3, pp. 242–257, 2000.
- [6] T. Kohonen, *Self-Organization and Associative Memory*, Springer, Berlin, Germany, 2nd edition, 1988.
- [7] J. Koza, *Genetic Programming: On the Programming of Computers by Means of Natural Selection*, MIT Press, Cambridge, Mass, USA, 1992.
- [8] N. Margolus, “CAM-8: a computer architecture based on cellular automata,” in *Pattern Formation and Lattice-Gas Automata*, A. Lawniczak and R. Kapral, Eds., pp. 167–187, American Mathematical Society, Providence, RI, USA, 1994.
- [9] J. A. G. Nijhuis, M. H. ter Brugge, K. A. Helmholz, et al., “Car license plate recognition with neural networks and fuzzy logic,” in *Proceedings of IEEE International Conference on Neural Networks Conference*, vol. 5, pp. 2232–2236, Perth, Australia, November–December 1995.
- [10] R. Poli, “Sub-machine-code GP: new results and extensions,” in *Proceedings of the 2nd European Workshop on Genetic Programming (EuroGP '99)*, R. Poli, P. Nordin, W. B. Langdon, and T. Fogarty, Eds., vol. 1598 of *Lecture Notes on Computer Science*, pp. 65–82, Springer, Göteborg, Sweden, May 1999.
- [11] R. Poli and W. B. Langdon, “Sub-machine-code genetic programming,” in *Advances in genetic programming: Volume 3*, chapter 13, pp. 301–323, MIT Press, Cambridge, Mass, USA, 1999.
- [12] T. Toffoli and N. Margolus, *Cellular Automata Machines: A New Environment for Modeling*, MIT Press, Cambridge, Mass, USA, 1987.

- [13] M. Tomassini and E. Sanchez, *Towards Evolvable Hardware*, Springer, Berlin, Germany, 1996.
- [14] J. von Neumann, *Theory of Self-Reproducing Automata*, University of Illinois Press, Urbana, Ill, USA, 1966.
- [15] S. Wolfram, *Theory and Applications of Cellular Automata*, World Scientific, Singapore, 1986.
- [16] D. H. Wolpert, “Stacked generalization,” *Neural Networks*, vol. 5, pp. 241–259, 1992.
- [17] D. Zongker and B. Punch, *lil-gp 1.01 user’s manual*, Michigan State University, 1996, <ftp://garage.cse.msu.edu/pub/GA/lilgp>.

Stefano Cagnoni: Dipartimento di Ingegneria dell’Informazione, Università di Parma,
Viale Usberti 181a, 43100 Parma, Italy

Email: cagnoni@ce.unipr.it

Monica Mordini: Dipartimento di Ingegneria dell’Informazione, Università di Parma,
Viale Usberti 181a, 43100 Parma, Italy

Email: monica@ce.unipr.it

Giovanni Adorni: Dipartimento di Informatica, Sistemistica e Telematica (DIST),
Università di Genova, Via all’ Opera Pia 13, 16145 Genova, Italy

Email: adorni@unige.it

4

Halftone image generation using evolutionary computation

Kiyoshi Tanaka and Hernán Aguirre

4.1. Introduction

In this chapter, we focus on halftone image generation using evolutionary computation (EC). Image halftoning is an important technique in the printing and display industry, in which an N -gray tone image must be properly portrayed as an n -gray tone image, where $n < N$. It is well known that a good halftone image satisfies both gray level precision and spatial resolution without including particular pixel patterns. However, since n is a limited (small) number, it is difficult to generate halftone image satisfying these requirements simultaneously. So far, various approaches have been developed such as ordered dithering, error diffusion, blue noise, and so on [34], but each scheme has its own advantages and disadvantages. For further improvement, a new attempt that uses genetic algorithms (GAs) to solve such complex image halftoning problem has been reported in two ways. One approach seeks to evolve filters, which are applied to the input N -gray tone image to generate a halftone image [8, 9, 32]. Another approach searches directly for the optimum halftone image having a visually similar appearance to the input N -gray tone image. The latter approach is interesting in the sense that the halftone image itself is directly represented as genetic information and is evolved by evaluation functions designed to generate desirable output images. From this point of view, here we focus on the latter approach.

Kobayashi and Saito [23, 24] first proposed a direct search GA-based halftoning technique to generate bilevel halftone images. This scheme divides the input images into nonoverlapping blocks and uses a simple GA [17, 19] with a specialized two-dimensional crossover to search the corresponding optimum binary patterns. The method's major advantages are that (i) it can generate images with a specific desired combination of gray level precision and spatial resolution, and (ii) it generates bilevel halftone images with quality higher than conventional schemes [34]. In this chapter, we will first explain this basic scheme, and then present some improved and extended schemes of this approach mainly for the reduction of

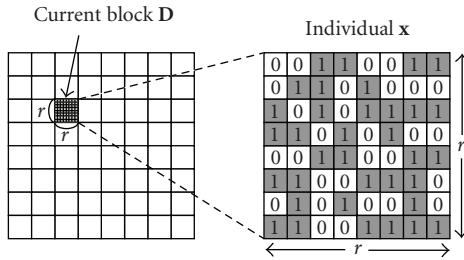


FIGURE 4.1. Image division and individual representation ($r \times r = 8 \times 8$).

computational cost and memory configuration. Finally, we will give some conclusions and future work.

4.2. Image halftoning scheme using GA (basic approach)

4.2.1. Individual representation

An input image is first divided into nonoverlapping blocks \mathbf{D} consisting of $r \times r$ pixels to reduce the search space of solutions [23, 24]. The GA uses an individual \mathbf{x} with an $r \times r$ two-dimensional representation for the chromosome. In case of bilevel halftoning, each element of the chromosome $x(i, j)$ ($i, j = 0, 1, \dots, r - 1$) $\in \{0, 1\}$. Figure 4.1 illustrates the image division into blocks and an example of individual \mathbf{x} corresponding to a current block \mathbf{D} .

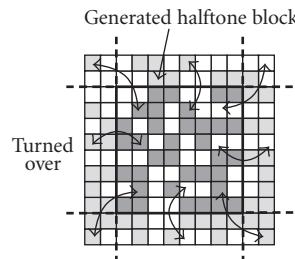
4.2.2. Evaluation

Chromosomes are evaluated with two kinds of evaluation criteria. (i) One is high gray level precision (local mean gray levels close to the original image), and (ii) the other is high spatial resolution (appropriate contrast near edges) [23, 24]. First we calculate a gray level precision error by

$$E_m = \sum_{(i,j) \in D} \frac{1}{r^2} |g(i, j) - \hat{g}(i, j)|, \quad (4.1)$$

where $g(i, j)$ ($i, j = 0, 1, \dots, r - 1$) is the gray level of the (i, j) th pixel in the input image block \mathbf{D} , and $\hat{g}(i, j)$ is the estimated gray level associated to the (i, j) th pixel of the generated halftone block ($x(i, j)$). To obtain $\hat{g}(i, j)$, a reference region around $x(i, j)$ is convoluted by a Gaussian filter that models the correlation among pixels. An example of a 5×5 filter is shown in Figure 4.2. In order to reduce discontinuity around block boundaries, the pixel pattern of \mathbf{x} is copied around the boundary regions as shown in Figure 4.3, and used to calculate the gray level estimation $\hat{g}(i, j)$.

1	4	7	4	1
4	20	33	20	4
7	33	54	33	7
4	20	33	20	4
1	4	7	4	1

FIGURE 4.2. An example of a 5×5 Gaussian filter.FIGURE 4.3. Discontinuity reduction by copying binary pattern of a current generated block \mathbf{x} around block boundaries.

In order to preserve the edge information of the input image well, we calculate the spatial resolution error by

$$E_c = \sum_{(i,j) \in D} \frac{1}{r^2} |G(i,j) - B(i,j)|, \\ G(i,j) = g(i,j) - \bar{g}(i,j), \\ B(i,j) = \left(x(i,j) - \frac{1}{2} \right) N, \quad (4.2)$$

where $G(i,j)$ is the difference between the gray level $g(i,j)$ and its neighboring local mean value $\bar{g}(i,j)$. $\bar{g}(i,j)$ is calculated with a 5×5 local average filter having uniform coefficients, and N denotes the dynamic range of input image.

These two errors E_m and E_c are combined into one single objective function as

$$E = \alpha_m E_m + \alpha_c E_c, \quad (4.3)$$

where α_m and α_c are weighting parameters of E_m and E_c , respectively. The chromosome's fitness is assigned by

$$F = E_{\max} - E, \quad (4.4)$$

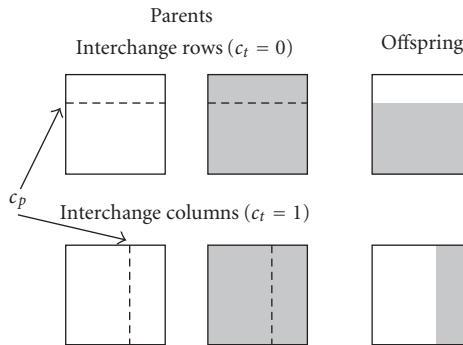
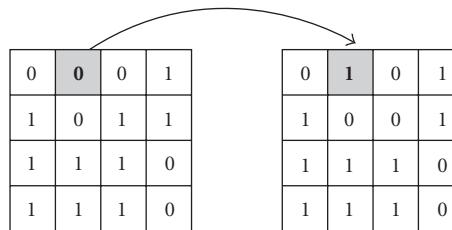


FIGURE 4.4. Illustration of two-dimensional crossover.

FIGURE 4.5. Illustration of bit-flipping mutation (block size is 4×4 pixels).

where E_{\max} is the error associated with the worst chromosome in a population. Using E_{\max} helps to induce a better scaling among solutions in order to assign selection probabilities, especially during the latest stages of the search. The GA is used to search for an optimum compromise between grey level precision and spatial resolution with the above fitness function.

4.2.3. Genetic operators and selection

Since we operate two-dimensional image data, crossover [17, 19] is implemented for two-dimensional chromosomes. Two random numbers, c_t and c_p , define its method of operation. First, $c_t = N[0, 1]$ is sampled to decide whether to interchange chromosomes' rows or columns from two selected parents, say (i) if $c_t = 0$, interchange rows and (ii) if $c_t = 1$, interchange columns. Then, $c_p = N[0, r]$ indicates the crossing point as shown in Figure 4.4. Both c_t and c_p are sampled new for each individual created by crossover. Although crossover can potentially create two offspring at a time, only one of them is randomly selected in this scheme.

After crossover, mutation inverts bits with a small probability per bit, p_m , analogous to canonical GA [17, 19]. For every bit actually selected for mutation, 0 becomes 1 and vice versa, as shown in Figure 4.5. In the following sections, we call the application of crossover followed by mutation as "CM."

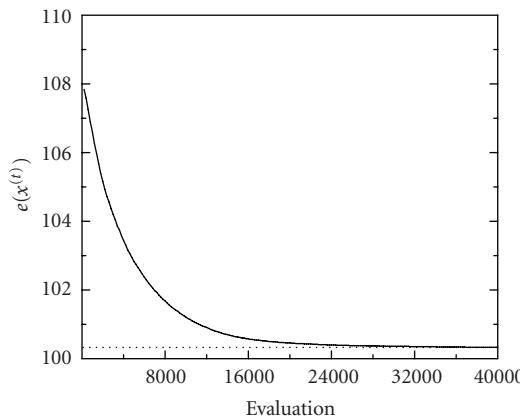


FIGURE 4.6. Average error transition over evaluation numbers for “Lenna.”

After offspring creation by genetic operators, proportional selection [17, 19] is applied for all individuals in the population $P(t)$ to select parent individuals for the next generation.

4.2.4. Results and discussion for basic scheme

4.2.4.1. Experimental setup

Through this chapter, to verify the performance of the schemes explained, we mainly use “Lenna” in SIDBA (Standard Image DataBase) as benchmark image. The size of the original images is 256×256 pixels with $N = 256$ gray levels and the generated images are bilevel halftone images ($n = 2$). The image block size is $r \times r = 16 \times 16$ and the population size is $\lambda = 200$ (200 offspring are created from 200 parents). The weighting parameters in (4.3) are set to $\alpha_m = 0.2$ and to $\alpha_c = 0.8$, which are suggested for an appropriate balance between E_m and E_c by visual assessment [23, 24]. The crossover and mutation rates are set to $p_c = 1$ and $p_m = 0.001$, respectively.

4.2.4.2. Error transition and generated images

Figure 4.6 shows the image’s average-error transition over evaluation numbers, which is calculated as the average of the best individuals’ error in all (256 pieces) image blocks. From the results, it can be seen that the error gradually converges to a constant value by spending more than 32 000 evaluations. The error value of (4.3) achieved by canonical GA (cGA) [17, 19] with the above settings after $T = 40\,000$ evaluations will be used as a reference value for image quality in the following sections. This value is shown as a broken line in the figure.

Figure 4.7 shows the original image “Lenna” and several generated halftone images by traditional methods and the GA-based scheme. First, we show images



FIGURE 4.7. Original and generated halftone images ("Lenna").

generated by conventional ordered dithering and error diffusion in Figures 4.7(b) and 4.7(c), respectively. We can see that image quality achieved by ordered dithering (Bayer matrix [34]) is insufficient with particular patterns caused by the periodic use of threshold matrix and low gray level precision. Error diffusion (Javis matrix [34]) fairly improves image quality but still has a few problems, that is, there are missing dots in high-light regions and particular patterns like worms can be clearly seen. On the other hand, a halftone image generated by cGA(200) with $T = 40\,000$ evaluations in Figure 4.7(d), which gives the image quality reference value, is quite smooth and less prone to particular patterns having both high gray level precision and high spatial resolution. Table 4.1 shows the values of the two kinds of errors, E_m and E_c , and the combined error E obtained for each method. From this table, we can see that both gray level precision and special resolution errors are remarkably reduced by using GA compared to the errors obtained by the conventional methods. By observing the difference with the error scale in Figure 4.6, we can see the significance of improvement by GA's optimization.

4.3. Accelerated halftoning scheme using improved GA

While the basic approach using a simple GA [23, 24] generates bilevel halftone images with quality higher than conventional techniques, it uses a substantial amount of computer memory and processing time that deprives it from practical implementations. In order to solve these drawbacks, in this section, we present

TABLE 4.1. Error values for each method.

errors	E_m	E_c	E
Ordered dithering (Bayer type)	10.16	126.05	102.87
Error diffusion (Javis matrix)	10.53	124.24	101.49
Canonical GA	10.04	122.72	100.19

an accelerated image halftoning scheme using an improved GA (called GA-SRM) with tiny populations [1, 2]. This scheme can generate high-quality images like the basic scheme, but reduces computer memory and processing time simultaneously.

4.3.1. Cooperative model for genetic operators

The improved GA-SRM [3, 4] is based on a model that puts crossover and varying mutation operators in a cooperative stand with each other. The model uses two operators applied in parallel (concurrently) to produce offspring. One is crossover followed by conventional “background” mutation (CM) and the other one is a varying mutation operator called self-reproduction with mutation (SRM). In addition, the model uses an extictive selection mechanism.

In CM, mutation is applied with small rate, therefore the amount of diversity introduced by mutation is modest. For the same reason, the disruption that mutation causes to crossover in CM is also expected to be small. On the other hand, varying mutation SRM uses higher mutation rates and is applied parallel to CM, avoiding interferences between crossover and high mutation. Thus, high mutations when harmful will have a negative impact on the *propagation* of beneficial recombinations already present in the parent population. However, it will not affect the *creation* of beneficial recombinations by crossover. Likewise, in the case that crossover produces poor performing individuals it would not affect the survivability of beneficial mutations introduced by SRM that can contribute to the search. The explicit parallel formulation of CM and SRM gives an efficient framework to achieve better balances for mutation and crossover during the run of the algorithm, in which the strengths of higher mutation and crossover can be kept without interfering one with the other.

The parallel formulation of CM and SRM can avoid interferences between crossover and high mutation; however, it cannot prevent SRM from creating deleterious mutations or CM from producing ineffective crossing over operations. To cope with these cases, the model also incorporates the concept of extictive selection that has been widely used in evolution strategies. Through extictive selection the offspring created by CM and SRM coexist competing for survival (the poor performing individuals created by both operators are eliminated) and reproduction. The block diagram of this model is shown in Figure 4.8.

4.3.2. Genetic operators

In the improved scheme [1, 2], we follow the individual representation and evaluation functions used in the basic approach explained in Section 4.2. Also, to

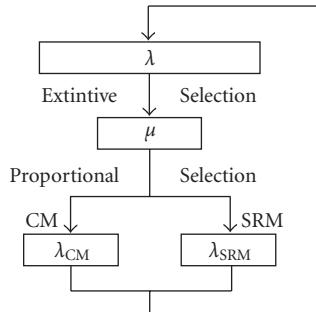


FIGURE 4.8. Block diagram of improved GA (GA-SRM).

produce offspring with CM, we use the same two-dimentional crossover followed by mutation with small probability $p_m^{(CM)}$ as explained in Section 4.2.3. However, we introduce a new varying mutation genetic operator SRM and modify selection. We explain SRM in the following and the selection in the next subsection.

To produce offspring with SRM, first an individual is selected from the parent population $P(t)$, an exact copy is created and then mutation is applied only to the bits inside a mutation block. SRM is provided with an adaptive dynamic-block (ADB) mutation schedule similar to adaptive dynamic-segment (ADS) mutation in [3, 4].

With ADB, mutation is directed only to a block (square region) of the chromosome and the mutation block area $\ell \times \ell$ is dynamically adjusted (decreases) every time a normalized mutants survival ratio falls under a threshold, $\gamma < \tau$, as shown in Figure 4.9. The offset position of the mutation block for each chromosome is chosen at random. The normalized mutant survival ratio is specified by

$$\gamma = \frac{\mu_{SRM}}{\lambda_{SRM}} \cdot \frac{\lambda}{\mu}, \quad (4.5)$$

where μ is the number of individuals in the parent population $P(t)$, μ_{SRM} is the number of individuals created by SRM present in $P(t)$ after selection, λ_{SRM} is the offspring number created by SRM, and λ is the total offspring number, $\lambda_{CM} + \lambda_{SRM}$ (see Figure 4.8).

Two kinds of mutation schemes are investigated for ADB: (i) quantitative and (ii) qualitative mutation. Quantitative mutation in ADB is implemented as the standard bit-flipping process as shown in Figure 4.5. Mutation probability for the bits inside the segment is $p_m^{(SRM)} = \alpha$. After this kind of mutation has been applied, the contrast near edges and the local mean average might change in an individual affecting both E_c and E_m in (4.3).

On the other hand, qualitative mutation in ADB is implemented as a bit-swapping process. First, a set B containing the indexes of all the bits in the mutation block is initialized. Next, a pair of indexes in B corresponding to bits b' and b'' are randomly marked and then swapped in the mutation block as shown in

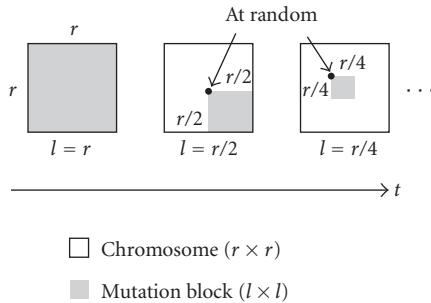


FIGURE 4.9. Adaptive dynamic-block (ADB) reduction. Mutation is directed only to the $\ell \times \ell$ shaded region of the chromosome.

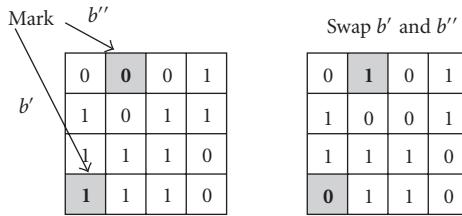


FIGURE 4.10. Illustration of bit-swapping mutation (block size is 4×4 pixels).

Figure 4.10. The marked indexes are removed from B and the marking-swapping process is repeated until there are no remaining indexes in B . Note that it is not necessary to set a mutation probability in qualitative mutation since all pairs of bits within the mutation block are simply swapped.

Also note that after qualitative mutation, the number of 0's and 1's remains unchanged. In other words, qualitative mutation has an impact only on the evaluation of the spatial resolution's error, E_c , but not on the gray level resolution's error, E_m , in (4.3). This kind of mutation could take better advantage of the high correlation among contiguous pixels in an image [18], and contribute to a more effective search.

4.3.3. Selection

(μ, λ) proportional selection [11] implements the required extictive selection mechanism. Selection probabilities are computed by

$$p(\mathbf{x}_k^{(t)}) = \begin{cases} \frac{f(\mathbf{x}_k^{(t)})}{\sum_{l=1}^{\mu} f(\mathbf{x}_l^{(t)})} & (1 \leq k \leq \mu), \\ 0 & (\mu < k \leq \lambda), \end{cases} \quad (4.6)$$

where $\mathbf{x}_k^{(t)}$ is an individual at t th generation which has the k th highest fitness value $f(\mathbf{x}_k^{(t)})$ in $P(t)$, μ is the number of parents, and λ is the number of offspring. Also,

we assure that the two parents selected for crossover are different, $\mathbf{x}_k^{(t)}$ and $\mathbf{x}_l^{(t)}$ ($k \neq l$). Note that with this kind of selection, we can easily control selection pressure by varying the value of μ . Setting $\mu = \lambda$ we have conventional proportional selection and by reducing the value of μ we increase selection pressure.

4.3.4. Experimental results and discussion

4.3.4.1. Experimental setup

To verify the performance of the improved scheme, here we use “Girl” (256×256 pixels with 256 gray levels) in SIDBA. The weighting parameters in (4.3) and the block size are set to the same values used in Section 4.2. For each block, the algorithm was ended after the same total evaluation numbers $T = 4 \times 10^4$ (the number of generations is calculated as T/λ in this scheme). Mutation probability for CM is set accordingly to $p_m^{(\text{CM})} = 0.001$. $\lambda_{\text{CM}} : \lambda_{\text{SRM}} = 1 : 1$ for offspring creation and $\mu : \lambda = 1 : 2$ (extinctive pressure), which is proved to be the best parameters’ balance for a robust and reliable search [3, 4]. Also, we set $\tau = 0.40$ as a threshold for the normalized mutant survival ratio specified by (4.5). Mutation probability for ADB with quantitative mutation is $p_m^{(\text{SRM})} = 0.125$. In case of qualitative mutation, it is not necessary to set a mutation probability.

4.3.4.2. Performance comparison with basic scheme

To observe the performance by the improved scheme with GA-SRM, we set the population sizes to $\mu = \lambda_{\text{CM}} = \lambda_{\text{SRM}} = 100$. With these values it creates the same number of offspring (200 offspring from 100 parents) as the basic scheme with cGA does (200 offspring from 200 parents). Figure 4.11 shows the image’s average-error transition by the two schemes. From this figure, it can be seen that GA-SRM converges faster and reaches better quality levels than cGA. Also, as expected, qualitative mutation performs better than quantitative mutation. Under this population configuration, qualitative mutation (GA-SRMs) needs only $0.34T$ evaluations to surpass the final image quality levels obtained by cGA, whereas $0.7T$ evaluations are needed in case of quantitative mutation (GA-SRMf).

SRM’s behavior can be observed from Figure 4.12, which presents the block’s side length reduction, ℓ , and the number of individuals produced by SRM-ADB that survive selection, μ_{SRM} , for one image block. From this figure, it is clear that (i) SRM contributes with beneficial mutations (carried by mutants that survive selection) in every generation of the search process, and (ii) the key factor for SRM to be an effective operator lies in its own regulation mechanism: mutation block adjusted every time the number of mutants that survive selection falls under a minimum level τ .

4.3.4.3. Effect of population size reduction

Since GA-SRM introduces higher levels of diversity than cGA, we observe the performance of the algorithms with smaller populations where diversity is even a

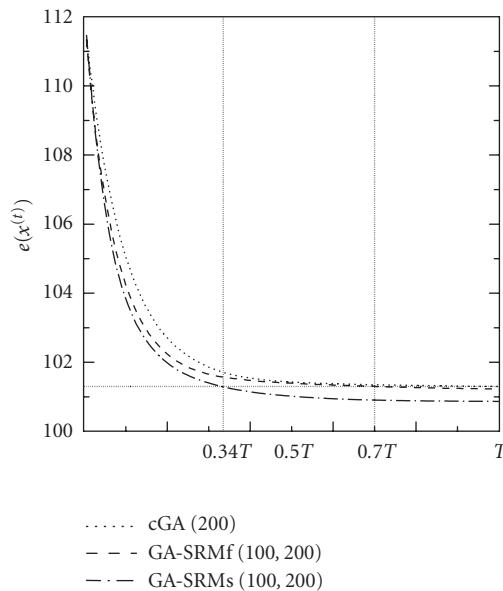


FIGURE 4.11. cGA and GA-SRMs performance using same size offspring population.

more important issue. Figure 4.13(a) shows results by cGA using $\{200, 100, 40, 20, 4\}$ population configurations. Figures 4.13(b) and 4.13(c) present results for equivalent configurations $\mu = \lambda_{CM} = \lambda_{SRM} = \{100, 50, 20, 10, 2\}$ by GA-SRMf and GA-SRMs, respectively, along with those obtained by cGA using a 200 population. From Figure 4.13(a), we can see that the 200 population size leads to the best image quality in cGA. As the population size is reduced, the final image quality is also deteriorated. Figure 4.13(b) shows that the introduction of quantitative mutation allows us to considerably reduce population sizes from 100 to 10 and still obtain a gain on search speed to generate images of quality similar or a little better compared to cGA. However, a further reduction in population sizes from 10 to 2 is not effective.

In Figure 4.13(c), we observe that GA-SRMs using qualitative mutation with smaller populations converge faster and always produce a better image quality than the one obtained by cGA. In this case, qualitative mutation not only allows to reach higher levels of image quality but also reduces the population configuration to its minimum level. This is because SRM with this kind of mutation always contributes to introduce diversity in levels such that SRM could be competitive with CM regardless of the population size, avoiding premature convergence, which is an important concern in cGA [17, 19]. GA-SRM's robust performance even with tiny populations allows us to choose the smallest memory configuration to generate halftone images without compromising the image quality. In fact, the GA-SRM using qualitative mutation with $\mu = 2$ and $\lambda = 4$ configuration (merely 2% of the population size used in basic scheme [23, 24]) attained after

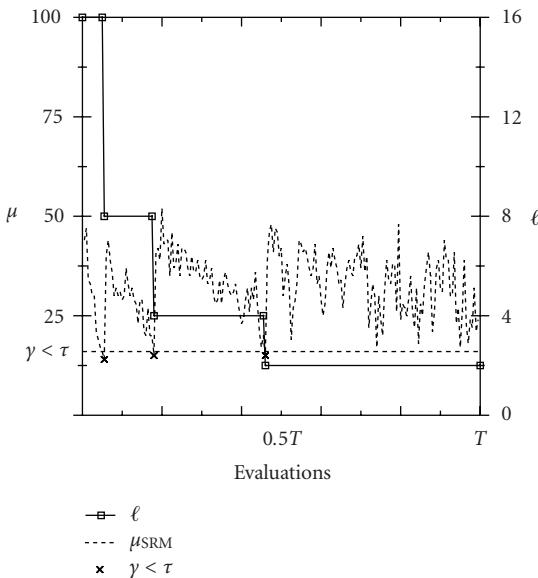


FIGURE 4.12. Mutation block's side length reduction and SRM-ADB offspring that survive selection.

only $0.15T$ evaluations the same image quality obtained by cGA after T evaluations.

4.3.4.4. Generated images

Figure 4.14 shows the original image “Girl,” the generated halftone image by cGA(200) after $0.15T$ evaluations and by GA-SRMs(2, 4) using qualitative mutation after $0.15T$ evaluations for visual comparison. Note that there is a notorious difference between (c) and (b), which is not sufficiently converged yet at this early stage.

4.4. Simultaneous halftone image generation with multiobjective GA

Image halftoning is a true multiobjective optimization (MO) problem, in which high gray level precision and high spatial resolution must be sought to achieve visually high-quality images. The appropriate combination of these two factors is not only device but also application dependent. Moreover, a combination that is appropriate for one image may not be the best for other, depending on the characteristics of the individual images. Hence, it is desirable to have a set of generated images where to choose from the images that best suit an application. The GA-based halftoning schemes explained before [1, 2, 23, 24], however, treat the problem as a single objective optimization problem by fixing the weighting parameters in (4.3), and can generate only one image at a time. Thus, to generate a set of images, these techniques must do it sequentially, one at a time.

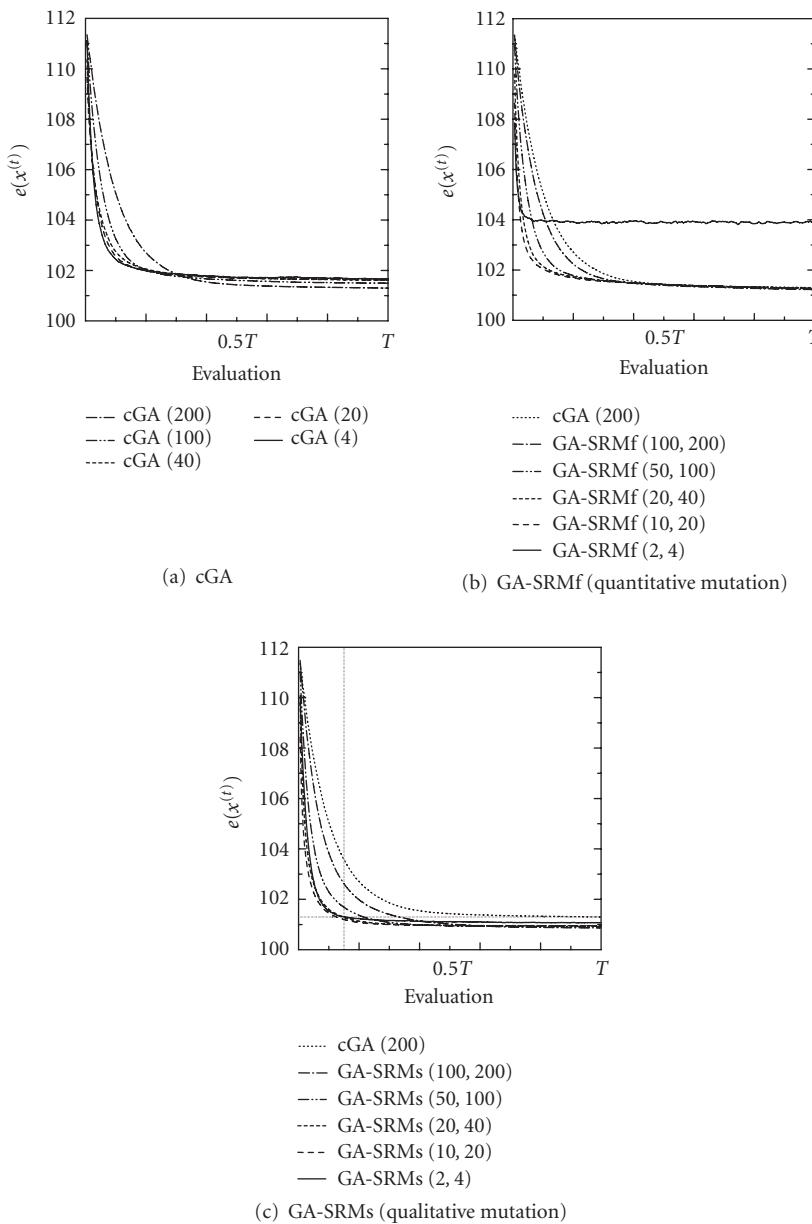


FIGURE 4.13. Performance comparison between cGA and GA-SRM.

In this section, we extend the improved halftoning scheme using GA-SRM [3, 4] to a multiobjective optimization GA [12, 14, 15, 20] and study its behavior and applicability generating simultaneously halftone images with various combinations of gray level precision and spatial resolution [5, 6].

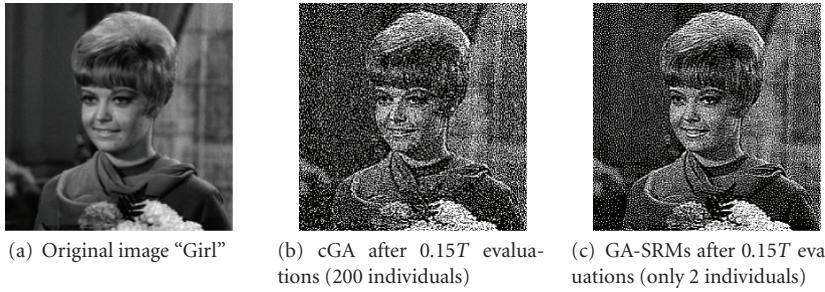


FIGURE 4.14. Original and generated halftone images ("Girl").

4.4.1. Multiobjective GA-SRM for halftoning problem

In order to extend GA-SRM [3, 4] to MO for halftone image generation, we follow a cooperative population search with aggregation selection [16, 25, 28, 33]. The population is monitored for nondominated solutions; however, Pareto-based fitness assignment is not directly used. A predetermined set of weights $\mathbf{W} = \{\omega^1, \omega^2, \dots, \omega^N\}$, which ponder the multiple objectives, defines the directions that the algorithm will search simultaneously in the combined space of the multiple objectives. N indicates the number of search directions. The n th search direction ω^n is a vector of nonnegative weights specified by $\omega^n = (\omega_1^n, \dots, \omega_M^n)$, where M indicates the number of objectives. The components in ω^n satisfy the conditions $\omega_m^n \geq 0$ ($m = 1, \dots, M$), and $\sum_{m=1}^M \omega_m^n = 1$.

4.4.2. Evaluation

We evaluate individuals by using the same two evaluation functions E_m and E_c in Section 4.2.2. Thus the number of objectives is $M = 2$. Normalized objective values, g_1 for E_m and g_2 for E_c , are assigned to each individual [5, 6].

The objective values are calculated once for each individual in the offspring population. However, we keep as many fitness values as search directions have been defined. A combined objective value is calculated for each search direction ω^n by

$$g^n(\mathbf{x}_i^{(t)}) = \sum_{m=1}^M \omega_m^n g_m(\mathbf{x}_i^{(t)}) = \omega_1^n g_1(\mathbf{x}_i^{(t)}) + \omega_2^n g_2(\mathbf{x}_i^{(t)}), \quad (4.7)$$

and the individuals' fitness in the n th search direction is assigned by

$$f^n(\mathbf{x}_i^{(t)}) = g^n(\mathbf{x}_W^{(t)}) - g^n(\mathbf{x}_i^{(t)}), \quad (4.8)$$

where $g^n(\mathbf{x}_W^{(t)})$ is the combined objective value associated with the worse individual in the n th search direction at the t th generation. Similar to (4.4), $g^n(\mathbf{x}_W^{(t)})$ helps to

induce a better scaling among solutions in order to assign selection probabilities, especially during the latest stages of the search.

4.4.3. Genetic operators and selection

For each search direction ω^n , CM creates a corresponding λ_{CM}^n number of offspring. Similarly, SRM creates λ_{SRM}^n offspring similar to the improved scheme [1, 2] explained in Section 4.2. Thus, the total offspring number for each search direction is $\lambda^n = \lambda_{CM}^n + \lambda_{SRM}^n$. The offspring created for all N search directions coexist within one single offspring population. Hence the overall offspring number is

$$\lambda = \sum_{n=1}^N \lambda^n. \quad (4.9)$$

SRM's mutation rates are adapted based on a normalized mutants survival ratio [1, 2], which is extended to

$$\gamma = \frac{\sum_{n=1}^N \mu_{SRM}^n}{\sum_{n=1}^N \lambda_{SRM}^n} \cdot \frac{\lambda}{\sum_{n=1}^N \mu^n}, \quad (4.10)$$

where μ^n is the number of individuals in the parent population of the n th search direction $P^n(t)$, μ_{SRM}^n is the number of individuals created by SRM present in $P^n(t)$ after extictive selection, λ_{SRM}^n is the offspring number created by SRM, and λ is the overall offspring number as indicated in (4.9).

Since we want to search simultaneously in various directions, selection to choose the parent individuals that will reproduce either with CM or SRM is accordingly applied for each one of the predetermined search directions. Thus, (μ, λ) proportional selection [11] is again applied for each search direction ω^n by

$$p^n(\mathbf{x}_k^{(t)}) = \begin{cases} \frac{f^n(\mathbf{x}_k^{(t)})}{\sum_{l=1}^{\mu^n} f^n(\mathbf{x}_l^{(t)})} & (1 \leq k \leq \mu^n \leq \lambda^n), \\ 0 & (\mu^n < k \leq \lambda), \end{cases} \quad (4.11)$$

where $\mathbf{x}_k^{(t)}$ is an individual at generation t which has the k th highest fitness value in the n th search direction $f^n(\mathbf{x}_k^{(t)})$.

Note that for each search direction, only $\lambda^n < \lambda$ individuals are created. However, the parent population μ^n is chosen among the overall λ offspring population. In this way, information sharing is encouraged among individuals created for neighboring search directions provided that the neighbors' fitness is competitive with the locals'. Figure 4.15 presents the block diagram of the extended multiobjective GA-SRM for the image halftoning problem.

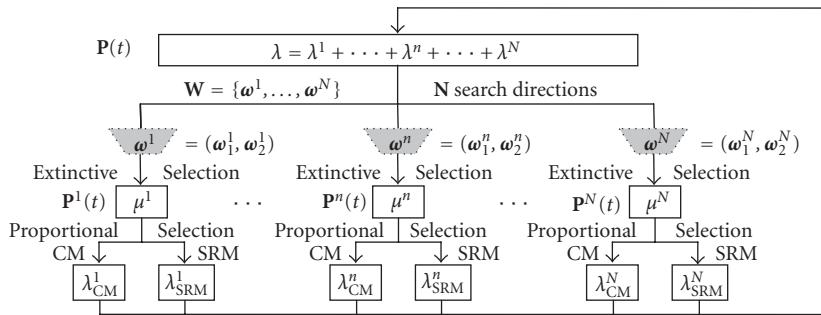


FIGURE 4.15. Block diagram of the extended multiobjective GA-SRM.

Once the offspring has been evaluated, a set of nondominated solutions is sought for each search direction, that is, for the n th search direction non-domination is checked only among the offspring created for that search direction. Two secondary populations keep the nondominated solutions. $P_{\text{cur}}(t)$ keeps the nondominated solution obtained from the offspring population at generation t and P_{nds} keeps the set of the nondominated solutions found through the generations. P_{nds} is updated at each generation with $P_{\text{cur}}(t)$. In the halftoning problem, an image is divided into blocks and the GA is applied to each image block. Hence, the GA would generate a set of nondominated solutions for each image block. Since we are interested in generating simultaneously various Pareto optimal “whole” images, a decision making process is integrated to choose only one solution for each search direction in each image block. Thus, among the various nondominated solutions found for a given search direction, we choose the one that minimizes the combined error E_m and E_c in that particular direction. Algorithm 4.1 illustrates the algorithm to simultaneously generate N halftone images with the extended multiobjective GA-SRM.

4.4.4. Experimental results and discussion

4.4.4.1. Experimental setup

We observe and compare the performance by four kinds of GAs: (i) a simple GA explained in Section 4.2 [23, 24] (denoted as cGA) (ii) an extended cGA using the same multiobjective technique described in this section (denoted as moGA), (iii) a GA-SRM explained in Section 4.3 [3, 4] (denoted as GA-SRM), and (iv) the extended multiobjective GA-SRM (denoted as moGA-SRM). These algorithms are applied to “Lenna.” For each image block, the algorithms were set with different seeds for the random initial population.

We define 11 search directions, $N = 11$, setting $\mathbf{W} = \{\omega^1, \omega^2, \dots, \omega^{11}\} = \{(0, 1), (0.1, 0.9), \dots, (1, 0)\}$ between E_m (gray level precision) and E_c (spatial resolution). With $\omega^1 = (0, 1)$, the search focuses exclusively in E_c ’s space and with

```

moGA-SRM procedure
begin
  divide original image into blocks
  set  $N$  search directions  $\mathbf{W} = \{\omega^1, \dots, \omega^N\}$ 
  for (each image block  $\mathbf{B}_u$ )
     $t = 0$ 
    initialize ( $P(0)$ )
    mo_evaluation ( $P(0)$ )
    while (not termination condition)
      for (each search direction  $\omega^n$ )
         $P^n(t) = (\mu, \lambda)$  proportional selection ( $P(t)$ )
         $P(t+1)^+ = \text{CM}(P^n(t))$ 
         $P(t+1)^+ = \text{SRM}(P^n(t))$ 
      done
      mo_evaluation ( $P(t+1)$ )
      get  $P_{\text{cur}}(t+1)$  from  $P(t+1)$ 
      update  $P_{\text{nds}}$  with  $P_{\text{cur}}(t+1)$ 
       $t = t + 1$ 
    done
     $\mathbf{G}_u = P_{\text{nds}}$ , keep  $N$  generated block images from  $\mathbf{B}_u$ 
  done
  generate  $N$  images ( $\mathbf{G}_u$ )
end

```

ALGORITHM 4.1. Algorithm to simultaneously generate N halftone images with the extended multiobjective GA-SRM.

$\omega^{11} = (1, 0)$ in E_m 's; whereas with ω^n , $2 \leq n \leq 10$, the search focuses in the combined space of E_c and E_m . moGA and moGA-SRM generate simultaneously 11 images, one image for each direction, in a single run. On the other hand, to generate 11 images with either cGA or GA-SRM, an equal number of separate runs are carried out, each one using a different ω^n as weighting parameter. The parameters used are summarized in Table 4.2.¹

4.4.4.2. Comparison between single and multiobjective GAs

Table 4.3 shows under column \mathbf{W} the average in all image blocks of the nonnormalized combined errors $e^n(\mathbf{x}) = \omega_1^n E_m(\mathbf{x}) + \omega_2^n E_c(\mathbf{x})$ by cGA(200) after $T = 40\,000$ evaluations for each search direction ω^n ($1 \leq n \leq 11$). For other algorithms under \mathbf{W} , we present the fraction of T at which the algorithms reach similar image quality (these values are all 1 for cGA(200) and are shown below the combined error). Column $T^{\mathbf{W}}$ indicates the overall evaluations needed to generate 11 images. Since the cGA generates one image at a time, it needs $11T$ evaluations to generate all 11 images.

¹GA-SRM search only in one direction at a time and the population related parameters μ^n , λ^n , λ_{CM}^n , and λ_{SRM}^n should be read without the index n .

TABLE 4.2. Genetic algorithms parameters.

Parameter	<i>cGA</i>	<i>moGA</i>	GA-SRM	<i>moGA-SRM</i>
selection	Proport.	(μ, λ) Proport.	(μ, λ) Proport.	(μ, λ) Proport.
Mating	$(\mathbf{x}_i, \mathbf{x}_j), i \neq j$			
p_c	0.6	0.6	1.0	1.0
$p_m^{(CM)}$	0.001	0.001	0.001	0.001
$\mu^n : \lambda^n$	—	1 : 1	1 : 2	1 : 2
$\lambda_{CM}^n : \lambda_{SRM}^n$	—	—	1 : 1	1 : 1

TABLE 4.3. Evaluations needed to generate high-quality images by *cGA*(200) for “Lenna.”

Algorithm	$\mathbf{W} = \{\omega^1, \omega^2, \dots, \omega^{11}\}$						T^W
	ω^1	ω^2	ω^4	ω^6	ω^9	ω^{11}	
Combined error	121.0	111.4	89.5	66.9	32.8	10.1	—
<i>cGA</i> (200)	1.00	1.00	1.00	1.00	1.00	1.00	$11T^\dagger$
<i>moGA</i> (18,198)	1.43	2.43	1.27	1.00	0.70	0.72	$2.43T^{\dagger\dagger}$
<i>moGA</i> (4,44)	1.12	2.30	1.36	1.02	0.73	0.79	$2.30T^{\dagger\dagger}$
GA-SRM(2,4)	0.40	0.23	0.13	0.11	0.09	0.08	$1.58T^\dagger$
<i>moGA-SRM</i> (9,198)	1.12	1.07	0.44	0.27	0.22	0.21	$1.12T^{\dagger\dagger}$
<i>moGA-SRM</i> (2,44)	1.56	1.03	0.30	0.16	0.12	0.12	$1.56T^{\dagger\dagger}$

† The entire number of evaluations required by the single objective GAs to generate all 11 images are given by the sum of the evaluations expended in each direction.

†† In the case of multiple objective GAs, due to the concurrent search, the maximum number of the evaluations among all search directions determines the overall number of evaluations needed to generate all 11 images.

The first *moGA* row shows results by the multiobjective simple GA with a $\mu^n = 18$ parents and a $\lambda^n = 18, \lambda = 198$ offspring configuration. *moGA* simultaneously generates 11 images and needs $2.43T$ to guarantee that the images in all search direction have at least the same quality as *cGA*(200). *moGA*’s second row shows results by *moGA* with a $\mu^n = 4$ parents and a $\lambda^n = 4, \lambda = 44$ offspring configuration. In this case, population size reduction in *moGA* accelerates a little bit more the overall convergence requiring $2.30T$ to produce better images than *cGA*(200). It should be noticed that population reductions in *cGA* accelerates convergence but it is affected by a loss of diversity and the final image quality is inferior than *cGA*(200)’s. *moGA* benefits from the information sharing induced by selection (see explanation in Section 4.4.4.4) and can tolerate population reductions. Compared with *cGA*, the results by *moGA* represent an enormous reduction in processing time and illustrates the benefits that can be achieved by including multiobjective techniques within GAs.

Row *GA-SRM*(2,4) presents results by *GA-SRM* with a 2-parent and 4-offspring configuration. *GA-SRM* even with a very scaled down population configuration considerably reduces processing time to sequentially generate high-quality

images for all combinations of weighting parameters. Compared with the $11T$ needed by cGA, GA-SRM needs only $1.58T$. Also, note that GA-SRM is faster than moGA.

The first moGA-SRM row shows results by the multiobjective GA-SRM with a $\mu^n = 9$ parents and a $\lambda^n = 18$, $\lambda = 198$ offspring configuration. Compared with moGA, we can see that the inclusion of SRM notoriously improves the multiobjective algorithm's performance requiring $1.12T$ to generate 11 images being faster than both GA-SRM and moGA. From GA-SRM and moGA-SRMs' results, we can see that the parallel mutation by SRM can greatly improve the performance of single as well as multiobjective genetic algorithms in the image halftoning problem.

Results by a scaled down population configuration is shown in row moGA-SRM(2,44) that represents a $\mu^n = 2$ parents and a $\lambda^n = 4$, $\lambda = 44$ offspring configuration. The population size reduction in moGA-SRM notoriously accelerates convergence in almost all the search directions. However, it delays convergence in ω^1 direction making the overall evaluation time to be slower than GA-SRM and moGA. This problem can be solved by dynamic configuration of computational resources (offspring creation between CM and SRM and evaluation numbers allocated to each search direction) [7].

4.4.4.3. Nondominated Pareto solutions

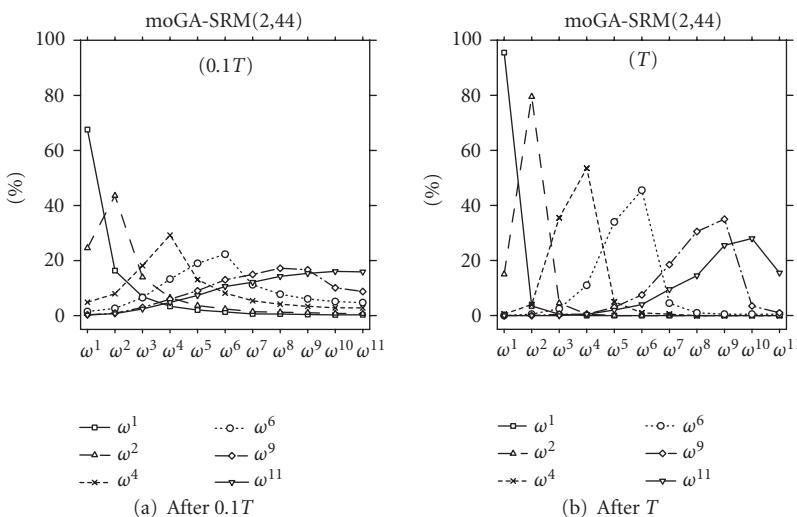
Our objective is to generate a set of strongly nondominated images for $N = 11$ predefined search directions. The generation of a set of images implies a three-step processes: (i) generation of nondominated solutions, (ii) clustering the solutions around the N search directions, and (iii) selection of the preferred solution for each search direction. In Table 4.4, under columns B_a and B_b , we present the preferred solutions obtained for each search direction in two typical image blocks. Column B_a illustrates a block in which the clusters are separated one from each other and the preferred solutions also form a strongly nondominated Pareto front. On the other hand, column B_b illustrates a block in which some clusters are very close one to another and the final preferred solution is the same in more than one search direction (see, e.g., ω^8 and ω^9 , or ω^{10} and ω^{11}). Also, from these two columns, we can see that the errors' ranges vary depending on the characteristics of the image block. Under whole images, we present the mean errors \bar{E}_m and \bar{E}_c on all image blocks of the assembled images for each search direction. We can see that in the average the proposed method induces a strongly nondominated Pareto front for the generated images.

4.4.4.4. Effect of information sharing

Figure 4.16 shows the average distribution of the parent population for some of the ω^n directions after $0.1T$ and T evaluations. For example, in Figure 4.16(a), the parent population of ω^4 is in average composed by 18% of individuals coming from ω^3 , 30% from ω^4 itself, and 13% from ω^5 . From these figures, we can see that each search direction benefits from individuals that initially were meant for

TABLE 4.4. Obtained Pareto front (“Lenna”).

W	Two typical image blocks				Whole images	
	B_a		B_b			
	E_m	E_c	E_m	E_c	\bar{E}_m	\bar{E}_c
ω^1	33.22	113.61	43.06	123.97	43.4	121.0
ω^2	26.67	113.71	16.48	124.35	21.0	121.3
ω^3	23.95	113.86	14.43	124.43	16.9	121.5
ω^4	16.22	114.87	13.65	124.58	12.3	122.1
ω^5	13.20	115.37	8.12	125.57	11.4	122.2
ω^6	13.19	115.41	7.86	125.72	9.8	122.7
ω^7	13.08	115.46	7.75	125.75	9.6	122.8
ω^8	10.11	118.36	7.53	125.93	9.4	123.5
ω^9	9.61	118.90	7.53	125.93	9.3	123.7
ω^{10}	9.52	119.04	7.05	126.06	9.2	123.8
ω^{11}	9.49	119.18	7.05	126.06	9.1	124.0

FIGURE 4.16. moGA-SRM’s average parent population distribution after $0.1T$ and T evaluations (“Lenna”).

other neighboring directions. This information sharing pushes forward the search reducing convergence times. Looking at Figures 4.16(a) and 4.16(b) we can see that the effect of information sharing is higher during the initial stages of the search.

Figure 4.17 shows some of the simultaneously generated halftone images by moGA-SRM(2,44) with dynamic configuration of computational resources [7] after $0.70T$. As can be observed, the images for each search direction are high-quality images and the difference in contrast and gray level precision can be visually appreciated.



FIGURE 4.17. Simultaneously generated halftone images by moGA-SRM(2,44) after $0.70T$.

4.5. Interblock evaluation method in GA-based halftoning scheme

The GA-based halftoning methods explained before evolve all image blocks independently from each other. A side effect of this is that the evaluation function becomes approximate for the pixels close to the boundaries between image blocks, which introduces false optima and delays the search. This affection becomes larger as we reduce block size. In this section, we present an interblock evaluation method to further reduce processing time (evaluation numbers) in the GA-based image halftoning technique [29, 30]. We design the algorithm to avoid noise in the fitness function by evolving all image blocks concurrently, exploiting the interblock correlation, and sharing information between neighbor image blocks.

4.5.1. Problem

Due to the expected high correlation between neighboring pixels in an image, the pixels copied around the boundaries of the generated block aim to reduce discontinuities between blocks (see Figure 4.3). However, these pixels are not true information of the generated neighbor blocks. Although mathematically the same fitness function is used for every pixel, from an information standpoint the conventional GA-based halftoning schemes induce a kind of approximate fitness function [21] for the pixels close to the boundary regions, which introduces false optima. This misleads the algorithm and greatly affects its search performance. Here we show examples of generated pixels and their reference region to calculate gray level estimation in Figure 4.18. Note that if the area of image block is reduced, the

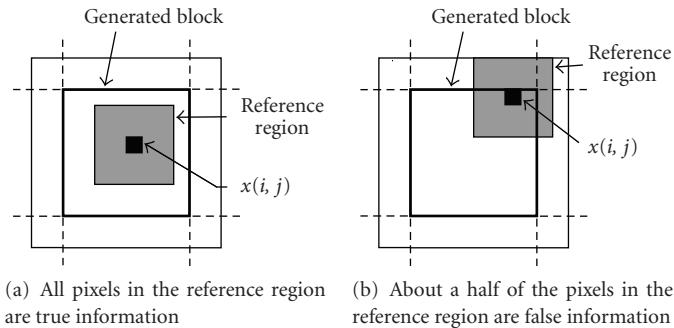


FIGURE 4.18. Examples of generated pixels and their reference region to calculate gray level estimation.

fraction of the number of pixels evaluated with the approximate function (e.g., Figure 4.18(b)) will increase while the fraction of the number of pixels evaluated with the true fitness function (e.g., Figure 4.18(a)) will decrease, negatively affecting the quality of the generated image and delaying the search. In addition, the ratio of true information used in the evaluation function reduces with the block size. For example, in case of $r \times r = 16 \times 16$ pixels, the ratio of true information used to calculate all $\hat{g}(i, j)$'s in a block becomes 0.8556, when the size of Gaussian filter is 5×5 . The ratio decreases to 0.7225 and 0.4900 as we reduce block size to $r \times r = 8 \times 8$ and 4×4 , respectively. The noise introduced by the approximated function becomes larger when we reduce block size, which is a real obstacle for further reduction of processing time.

4.5.2. Interblock evaluation method

To have a fitness function that models the halftoning problem with higher fidelity, we make use of the interblock correlation between neighbor blocks in the evaluation, linking each block with its neighbor blocks and sharing some genetic information between them [29, 30]. A GA is allocated to each block and each GA evolves its population of possible solutions concurrently. In this process, the best individuals $\mathbf{x}_{u,v}^{*(t-1)}$ in the neighbor populations are generationally referred and used to calculate the fitness values for individuals $\mathbf{x}_{k,l}^{(t)}$ ($k = 0, 1, \dots, K - 1$, $l = 0, 1, \dots, L - 1$) in the current population as shown in Figure 4.19. Here (k, l) denotes the address of the current block $\mathbf{D}_{k,l}$ in the input image, and (u, v) the address of linked neighbor blocks around $\mathbf{D}_{k,l}$.

Also, $*$ and t denote best individual and generation number, respectively. With this procedure of information sharing between populations we can supplement incomplete information in the evaluation process of [23, 24] expecting that it would contribute to reduce evaluation numbers, improve the image quality around block boundaries, and allow further reductions of block size. Parallel implementations can be realized with the required number of processing units, linking at most 8 neighbor units as illustrated in Figure 4.20. Here a unit $U_{k,l}$ ($k = 0, 1, \dots, K - 1$, $l = 0, 1, \dots, L - 1$) means a processor corresponding to the

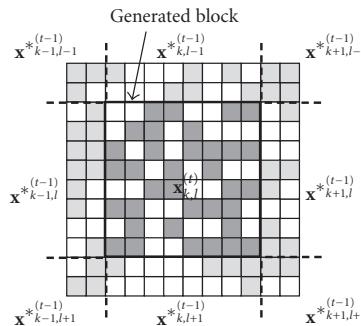


FIGURE 4.19. A current block and connected neighbor blocks for gray level estimation.

image block $\mathbf{D}_{k,l}$, which runs a GA to search the best binary pattern $\mathbf{x}_{k,l}^*$ for $\mathbf{D}_{k,l}$. In the following, we use an 8-neighbor topology as shown in Figure 4.20. All units evolve their populations and update best individuals generationally to interchange the information of reference pixels through the links. After completion of information sharing, all units synchronously start the evolution of next generation. Note that here the parallel GA is simulated as concurrent processes in a serial machine. That is, GA serially evolves all image blocks from the upper left $\mathbf{D}_{0,0}$ to the lower right $\mathbf{D}_{K-1,L-1}$ in a same generation and then we update reference pixels (best individuals $\mathbf{x}_{k,l}^{*(t)}$) for the next generation. Therefore, the following results show only the effects by the proposed interblock evaluation method.

4.5.3. Results and discussion

We apply this method to a canonical GA (cGA) [23, 24] and GA-SRM [1, 2]. To test the algorithms we use “Lenna” again. Parameters in cGA and GA-SRM use the same settings indicated in Sections 4.2 and 4.3, respectively.

4.5.3.1. Effects in conventional schemes

Table 4.5 shows the number of evaluations needed to reach the reference value for image quality in case that the block size is $r \times r = 16 \times 16$ pixels in cGA and GA-SRM. We can reduce the number of evaluations about 31% in cGA(200) and 16% in GA-SRM(100,200). Note that GA-SRM even with the conventional independent evaluation method is faster than cGA with the interblock evaluation method. Also, when we reduce population size in GA-SRM with the interblock evaluation method, it can be seen that smaller population sizes further accelerates the search without deteriorating the final image quality.

4.5.3.2. Effects in block size reduction

Next, we study the effect of reducing the size of the image block fixing the population size to $(\mu, \lambda) = (4, 8)$ in GA-SRM. Here, the mutation probability for CM is set

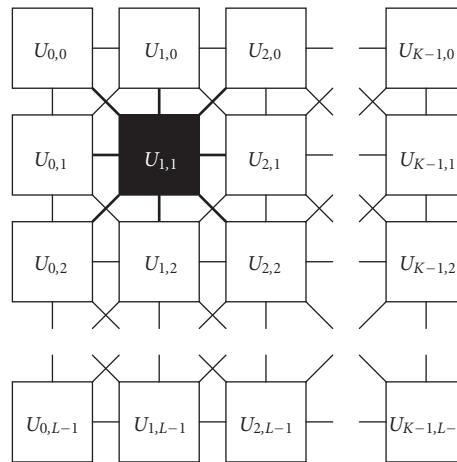


FIGURE 4.20. Parallel implementation with 8-neighbor topology.

TABLE 4.5. Comparison of the number of evaluations.

evaluation method	cGA	GA-SRM			
	(200)	(100,200)	(50,100)	(25,50)	(4,8)
independent	$1.000T$	$0.510T$	$0.330T$	$0.211T$	$0.115T$
interblock	$0.695T$	$0.430T$	$0.290T$	$0.185T$	$0.094T$

to $p_m^{(\text{CM})} = 1/r \times r$ [10], because this value for mutation rate causes better performance in combination with extictive selection [11]. Figure 4.21 plots the error transition over the evaluations, and Table 4.6 shows the number of evaluations needed to reach the image quality reference value. Note that with the interblock evaluation method we can further accelerate the search by reducing the block size to be evolved and still keep high-image quality. For example, in case of $r \times r = 4 \times 4$ the interblock evaluation method needs only 240 evaluations to achieve the image quality reference value (the same image quality obtained by cGA after 40 000 evaluations), which means less than 1/100 of the processing time compared with the basic scheme [23, 24]. We could consistently observe similar behavior for other benchmark images. At 0.006T, cGA(200) and GA-SRM(4,8) can never produce matured images as shown in Figures 4.22(a) and 4.22(b), but GA-SRM(4,8) with interblock evaluation method can produce high-quality halftone image by avoiding unpleasant noise around block boundaries caused by block-independent processing.

4.6. Summary

In this chapter, we described several image halftoning schemes using evolutionary computation (EC). We first explained the basic approach that uses a simple GA

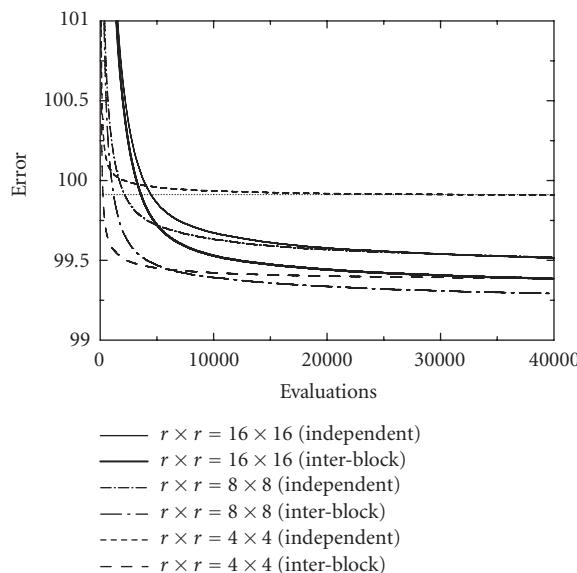


FIGURE 4.21. Performance by GA-SRM(4,8) with independent and interblock evaluation methods using different block sizes.

TABLE 4.6. Effect of block size reduction.

evaluation method	GA-SRM(4,8)		
	16 x 16	8 x 8	4 x 4
independent	0.112T	0.054T	0.84T
interblock	0.090T	0.029T	0.006T

to solve the halftoning problem, in which the input image is divided into small image blocks and the corresponding halftone block is generated by evolving chromosomes with two kinds of evaluation functions for (i) gray level precision and (ii) spatial resolution. This approach is promising in the sense that we can produce higher quality halftone images than conventional schemes such as ordered dithering, error diffusion, and so on. However, this scheme uses a substantial amount of computer memory and processing time that deprive it from practical implementations. To solve these drawbacks, next we presented an accelerated image halftoning scheme using an improved GA (GA-SRM) which uses two kinds of genetic operators, CM and SRM, and extictive selection. If we introduce adaptive dynamic block (ADB) reduction with qualitative mutation for SRM, we can drastically reduce memory size and processing time to generate halftone images without compromising the image quality. Only 2% of the population size and 15% of the evaluations were required to attain the same image quality obtained by the basic scheme. Third, we focused on the multiobjective nature of the image halftoning problem

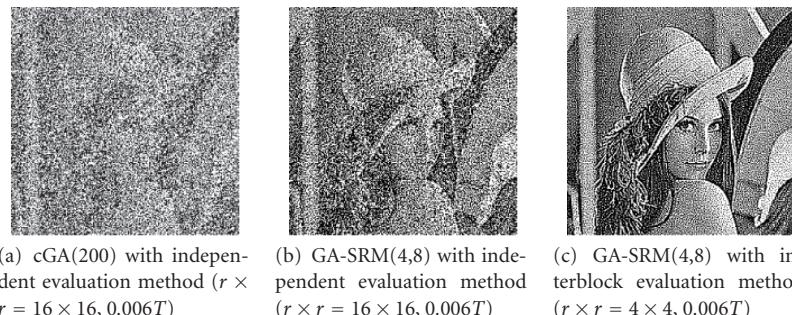


FIGURE 4.22. Comparison between generated halftone images at $T = 0.006$ evaluations.

to simultaneously generate halftone images having various combination of gray level precision and spatial resolution. The improved halftoning scheme using GA-SRM was extended to a multiobjective one for this purpose as well as to reduce total processing time. Consequently, we could reduce total processing time to 6% to generate simultaneously 11 halftone images with different weights for the two evaluation functions. Finally, we presented an interblock evaluation method to further reduce evaluation numbers in the GA-based image halftoning technique. We designed the algorithm to avoid noise in the fitness function by evolving all image blocks concurrently, exploiting the interblock correlation, and sharing information between neighbor image blocks. With this scheme, we could further reduce evaluation numbers to produce high-quality halftone images. Only 240 evaluations were required to surpass the reference value of image quality achieved by the basic scheme, which means only 0.6% of the total evaluation numbers required in the basic approach.

We mainly focused on the reduction of computational cost and memory configuration in GA-based halftoning schemes. However, several possibilities exist for further improvement and extensions that should be investigated. For example, this scheme can be extended to multilevel halftone image generation [35, 36], and color halftone image generation which is now being investigated by the authors. In case of color halftoning, evaluation functions should be properly modified by considering CMYK representation of colors for printing devices. Also, another possibility is information security for halftone images by digital watermarking [13, 22, 26, 27]. One approach [31] shares a signature image into two halftone images. In this method, the embedded secret image can be decoded by optically overlapping the two images generated for authentication and delivery. These are only a few trials among several possibilities and the authors are looking forward to further improvement and development of this research field.

Bibliography

- [1] H. E. Aguirre, K. Tanaka, and T. Sugimura, “Accelerated halftoning technique using improved genetic algorithm with tiny populations,” in *Proceedings of the IEEE International Conference on Systems, Man and Cybernetics*, vol. 4, pp. 905–910, Tokyo, Japan, October 1999.

- [2] H. Aguirre, K. Tanaka, and T. Sugimura, "Accelerated image halftoning technique using improved genetic algorithm," *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, vol. E82-A, no. 8, pp. 1566–1574, 2000.
- [3] H. E. Aguirre, K. Tanaka, and T. Sugimura, "Cooperative model for genetic operators to improve GAs," in *Proceedings of International Conference on Information Intelligence and Systems*, pp. 98–106, Bethesda, Md, USA, October–November 1999.
- [4] H. E. Aguirre, K. Tanaka, T. Sugimura, and S. Oshita, "Cooperative-competitive model for genetic operators: contributions of extictive selection and parallel genetic operators," in *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO '00)*, pp. 6–14, Las Vegas, Nev, USA, July 2000.
- [5] H. Aguirre, K. Tanaka, T. Sugimura, and S. Oshita, "Halftone image generation with improved multiobjective genetic algorithm," in *Proceedings of the 1st International Conference on Evolutionary Multi-Criterion Optimization*, vol. 1993 of *Lecture Notes in Computer Science*, pp. 501–515, Springer, Zurich, Switzerland, March 2001.
- [6] H. Aguirre, K. Tanaka, T. Sugimura, and S. Oshita, "Simultaneous halftone image generation with improved multiobjective genetic algorithm," *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, vol. E84-A, no. 8, pp. 1869–1882, 2001.
- [7] H. Aguirre, K. Tanaka, T. Sugimura, and S. Oshita, "Dynamic reconfigurations for simultaneous halftone image generation with multiobjective genetic algorithm," in *Proceedings of IEEE-EURASIP International Workshop on Nonlinear Signal and Image Processing*, Baltimore, Md, USA, June 2001.
- [8] J. T. Alander, T. Mantere, and T. Pyylampi, "Threshold matrix generation for digital halftoning by genetic algorithm optimization," in *Intelligent Robots and Computer Vision XVII: Algorithms, Techniques, and Active Vision*, vol. 3522 of *Proceedings of SPIE*, pp. 204–212, Boston, Mass, USA, November 1998.
- [9] J. T. Alander, T. Mantere, and T. Pyylampi, "Digital halftoning optimization via genetic algorithms for ink jet machine," in *Developments in Computational Mechanics with High Performance Computing*, pp. 211–216, CIVIL-COMP Press, Edinburg, UK, 1999.
- [10] T. Bäck, "Optimal mutation rates in genetic search," in *Proceedings of the 5th International Conference on Genetic Algorithms*, pp. 2–8, Morgan Kaufmann, San Mateo, Calif, USA, 1993.
- [11] T. Bäck and F. Hoffmeister, "Extended selection mechanism in genetic algorithms," in *Proceedings of the 4th International Conference on Genetic Algorithms*, pp. 92–99, Morgan Kaufmann, San Mateo, Calif, USA, 1991.
- [12] C. Coello, D. Veldhuizen, and G. Lamont, *Evolutionary Algorithms for Solving Multi-Objective Problems*, Kluwer Academic Publishers, Boston, Mass, USA, 2002.
- [13] I. Cox, M. Miller, and J. Bloom, *Digital Watermarking*, Morgan Kaufmann, San Mateo, Calif, USA, 2002.
- [14] K. Deb, *Multi-Objective Optimization Using Evolutionary Algorithms*, John Wiley & Sons, New York, NY, USA, 2001.
- [15] C. Fonseca and P. Fleming, "An overview of evolutionary algorithms in multiobjective optimization," *Evolutionary Computation*, vol. 3, no. 1, pp. 1–16, 1995.
- [16] M. P. Fourman, "Compaction of symbolic layout using genetic algorithms," in *Proceedings of the 1st International Conference on Genetic Algorithms (ICGA '85)*, pp. 141–153, Pittsburgh, Pa, USA, July 1985.
- [17] D. E. Goldberg, *Genetic Algorithms in Search, Optimization and Machine Learning*, Addison-Wesley, Reading, Mass, USA, 1989.
- [18] R. C. Gonzalez and R. E. Wood, *Digital Image Processing*, Addison-Wesley, Reading, Mass, USA, 1992.
- [19] J. H. Holland, *Adaptation in Natural and Artificial Systems*, University of Michigan, Ann Arbor, Mich, USA, 1975.
- [20] J. Horn, "Multicriterion decision making," in *Handbook of Evolutionary Computation*, vol. 1, pp. F1.9:1–F1.9:15, Oxford University Press, Oxford, UK, 1997.

- [21] Y. Jin, M. Olhofer, and B. Sendhoff, “A framework for evolutionary optimization with approximate fitness functions,” *IEEE Transactions on Evolutionary Computation*, vol. 6, no. 5, pp. 481–494, 2002.
- [22] S. Katzenbeisser and F. Petitcolas, *Information Hiding: Techniques for Steganography and Digital Watermarking*, Artech House, Boston, Mass, USA, 2000.
- [23] N. Kobayashi and H. Saito, “Halftoning technique using genetic algorithm,” in *Proceedings of IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP '94)*, vol. 5, Adelaide, SA, Australia, April 1994.
- [24] N. Kobayashi and H. Saito, “Halftoning technique using genetic algorithms,” *IEICE Transactions*, vol. J78-D-II, no. 10, pp. 1450–1459, 1996 (Japanese).
- [25] F. Kurwase, “A Variant of Evolution Strategies for Vector Optimization,” in *Proceedings of the 1st Workshop on Parallel Problem Solving from Nature*, vol. 496 of *Lecture Notes in Computer Science*, pp. 193–197, Springer, Dortmund, Germany, October 1990.
- [26] K. Matsui and K. Tanaka, “Video-steganography: how to secretly embed a signature in a picture,” in *Proceedings: Technological Strategies for Protecting Intellectual Property in the Networked Multimedia Environment*, vol. 1, pp. 187–206, Annapolis, Md, USA, January 1994.
- [27] K. Matsui, *Fundamentals of Digital Watermark*, Morikita Publishing, Tokyo, Japan, 1998.
- [28] T. Murata and H. Ishibuchi, “MOGA: multi-objective genetic algorithms,” in *Proceedings of the IEEE Conference on Evolutionary Computation (ICEC '95)*, vol. 1, pp. 289–294, Perth, WA, Australia, December 1995.
- [29] E. Myodo, H. Aguirre, and K. Tanaka, “Improved image halftoning technique using GAs with concurrent inter-block evaluation,” in *Proceedings of Genetic and Evolutionary Computation Conference (GECCO '03)*, vol. 2724 of *Lecture Notes in Computer Science*, pp. 2251–2262, Chicago, Ill, USA, July 2003.
- [30] E. Myodo, H. Aguirre, and K. Tanaka, “Inter-block evaluation method to further reduce evaluation numbers in GA-based image halftoning technique,” *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, vol. E87-A, no. 10, pp. 2722–2731, 2004.
- [31] E. Myodo and K. Tanaka, “A watermark sharing scheme to high quality halftone images with genetic algorithms,” in *Proceedings of the 6th European Workshop on Evolutionary Computation in Image Analysis and Signal Processing*, vol. 3005 of *Lecture Notes in Computer Science*, pp. 339–348, Coimbra, Portugal, April 2004.
- [32] J. L. Newbern and V. M. Bove Jr, “Generation of blue noise arrays by genetic algorithm,” in *Human Vision and Electronic Imaging II*, vol. 3016 of *Proceedings of SPIE*, pp. 441–450, San Jose, Calif, USA, February 1997.
- [33] J. Schaffer, “Multiple objective optimization with vector evaluated genetic algorithms,” in *Proceedings of the 1st International Conference on Genetic Algorithms*, J. Grefenstette, Ed., pp. 93–100, Hillsdale, NJ, USA, 1985 July.
- [34] R. Ulichney, *Digital Halftoning*, MIT Press, Cambridge, Mass, USA, 1987.
- [35] T. Umemura, H. Aguirre, and K. Tanaka, “Multi-level image halftoning technique with genetic algorithms,” *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, vol. E85-A, no. 8, pp. 1892–1897, 2002.
- [36] Y. Yoshizawa, H. Aguirre, and K. Tanaka, “Inter-block evaluation method in multi-level image halftoning technique using GA,” in *Proceedings of IEEE International Conference on Systems, Man and Cybernetics (SMC '04)*, vol. 3, pp. 3033–3039, The Hague, The Netherlands, 2004.

Kiyoshi Tanaka: Faculty of Engineering, Shinshu University, 4-17-1 Wakasato, Nagano 380-8553, Japan

Email: ktanaka@shinshu-u.ac.jp

Hernán Aguirre: Faculty of Engineering, Shinshu University, 4-17-1 Wakasato, Nagano 380-8553, Japan

Email: ahernan@shinshu-u.ac.jp

5

Evolving image operators directly in hardware

Lukáš Sekanina and Tomáš Martínek

5.1. Introduction

Some engineering positions, which a certain level of creativity is required for, will probably be replaced by computers in the near future. In this chapter, we will consider an engineer who develops image operators for real-world applications of image processing. In particular, he or she is responsible for designing low-level image filters and operators for smoothing, edge detection, or noise removal [16]. Let us assume that those filters will be utilized in an industrial application to preprocess images coming from, let us say, a camera. These images may be contaminated by a variety of noise sources (e.g., photon or on-chip electronic noise) and also by distortions such as shading, shots, or improper illumination. In order to perform the required preprocessing or to suppress the noise in a given application, a problem-specific filter has to be created. Traditionally, the engineers use a library of various filters and operators and manually tune promising variants of filters for the given application. In the process of tuning, various properties of filters might be optimized: coefficients, structure, size, power consumption, delay, and so forth. Assume that a convolution filter is applied. Therefore, we are interested in the spatial domain where an input image, x , convolves with a filter function, h . In discrete convolution, a kernel is shifted over the image and multiplies its values with the corresponding pixel values of the image. A kernel is a small matrix of coefficients; its members define weights of accounted pixels. The designer has to find these coefficients. He or she has to arrange a set of experiments in which candidate filters (typically developed intuitively) are evaluated on typical images from the application domain. The approach is based on an experimental work with large data sets. This type of work is usually time consuming even if a cluster of processors is used to evaluate candidate filters. The designer is mainly faced with the following problems.

- (i) How to invent the required filter which fulfills the objectives.
- (ii) How to evaluate the filter in a reasonably short time.

A print edition of this book can be purchased at

<http://www.hindawi.com/spc.8.html>

<http://www.amazon.com/dp/9774540018>

As the first problem primarily deals with a creative work, the second one requires a huge computational power. Candidate filters are evaluated using image databases collected from a given application domain of image processing (e.g., the data from production line inspection systems, traffic systems, biometric systems, microscopes, satellites, etc.). Those databases usually contain millions of images.

This chapter suggests replacing the creative engineer by an evolvable hardware system [5] which is able to invent the required filter automatically. Furthermore, rather than using a common PC or a cluster of workstations, a complete hardware implementation is developed for FPGA (field programmable gate array). Image filters are considered as digital circuits. The proposed approach has several advantages.

- (i) The proposed system is able to automatically design (evolve) an image filter if the corrupted image and the original image are supplied by user.
- (ii) As we a priory do not assume anything about the structure of the filter, novel (typically nonlinear) solutions might be evolved for each task. Thus we are not restricted to the class of well-known filters, such as mean filters, convolution filters, or median filters.
- (iii) As a complete hardware implementation is utilized, the time of evolution is much shorter in comparison with the time needed by an engineer to solve this problem. For some applications it could also be possible to perform a real-time (evolutionary) design in situ.
- (iv) The complete system can be implemented as a single chip or on a small board which is useful for embedded systems and industrial applications. Furthermore, the solution would be much cheaper than a PC-based solution.

The filters considered here will be composed of elementary blocks that are able to calculate the minimum, maximum, average, and some logic functions over two pixels. In order to establish a particular filter from those components, an extended variant of genetic programming, called functional-level Cartesian genetic programming (CGP) [10, 13], will be utilized. The approach will be demonstrated on typical tasks of image preprocessing, such as noise removal and edge detection. Since CGP represents a very simple algorithm it is very suitable for hardware implementation.

The main feature of the proposed implementation is that everything is implemented on a cutting-edge reconfigurable hardware platform available today. For experiments presented in this chapter we utilized the PCI COMBO6 card with Xilinx Virtex II FPGA developed in the Liberouter project [8]. Therefore, our results should indicate what is possible to achieve with an FPGA-based evolvable system nowadays. The evolutionary algorithm, implemented in hardware, is used to find a filter which minimizes the difference between the corrupted image and training image. These images are stored in RAM memories available on the COMBO6 card. A personal computer is used only for communication with the COMBO6 card, that is, for writing/reading the images to/from RAMs, and so forth.

5.2. Everything in hardware

Although a number of papers have dealt with the evolutionary image filter design at the hardware level (see, e.g., [1, 3, 4, 6, 11, 13–15, 19, 20]) none of them have introduced a *complete hardware implementation*. By *complete hardware implementation* we mean that the evolving filter as well as the evolutionary algorithm is implemented in hardware (i.e., in the FPGA in our case). The primary advantages of this approach are high speed, low cost, and potentially low-power consumption. The idea of complete hardware evolution for FPGAs was initially demonstrated by Tufte and Haddow [17]; however, they provided only a simple example of optimization of a few coefficients stored in a register.

Creating an application specific hardware, that is, a dedicated computer, can sometimes solve the problem of performance. A dedicated computer contains application-specific units and interconnecting networks that are not available in common processors. Considering the same microelectronic technology and operational frequency, a dedicated computer can be faster in several orders of magnitude than a universal computer for certain tasks [2]. Dedicated computers are produced as application-specific integrated circuits (ASIC). A high fabrication cost is the main disadvantage of ASICs. In most cases, millions of these specialized chips have to be manufactured to make the chip production commercially attractive.

General-purpose *reconfigurable devices* like field programmable gate arrays offer us other implementation options. The reconfigurable devices operate according to a configuration bitstream that is stored in an on-chip configuration memory. By writing to the configuration memory, the user can physically create new (digital) electronic circuits. An advantage of this approach is that a new hardware functionality is obtained through a simple reprogramming of the chip. Contemporary FPGAs contain thousands of programmable elements and operate at hundreds of MHz [18]. Figure 5.1 shows a typical architecture of the FPGA, which is a two-dimensional array of reconfigurable resources that include reconfigurable cells (e.g., 4-input look-up tables), reconfigurable interconnection network and reconfigurable I/O blocks. FPGA vendors provide collections of tools that are utilized for designing circuits for FPGAs. A designer describes a target (dedicated) computer in a hardware description language (such as VHDL). After simulations, a specialized program, a synthesizer, generates the circuit implementation and configuration bitstream for a given FPGA. The configuration bitstream is then uploaded into the configuration memory of the reconfigurable device.

An evolutionary algorithm will be used to find a configuration of an array of programmable elements in order to perform the filtering task. Candidate filters will be evaluated using a training image. These filters have to minimize the difference between a corrupted version and uncorrupted version of the training image. The corrupted version will contain a particular type of noise. We will show that the evolved filters work effectively for a reasonable class of test images corrupted by the same type of noise. A special filter will be evolved for a given type of noise.

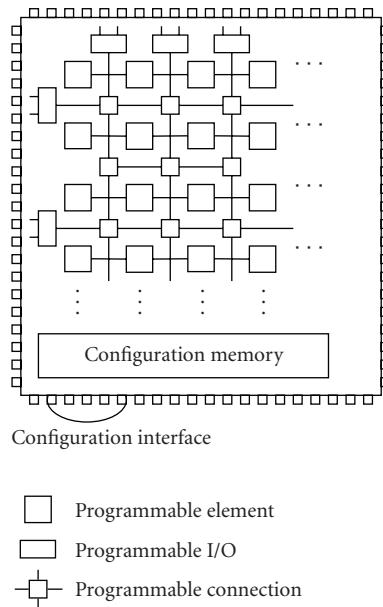


FIGURE 5.1. FPGA consists of programmable elements, programmable interconnection network, and programmable I/O ports. Its function is defined by uploading configuration bits into the configuration memory.

The evolvable image filter introduced in this chapter is composed of three main components—virtual reconfigurable circuit, fitness unit, and genetic unit. These components will be implemented on a single FPGA (see Figure 5.2).

5.2.1. Image filters in a virtual reconfigurable circuit

Every image operator will be considered as a digital circuit of nine 8-bit inputs and a single 8-bit output, which processes gray-scaled (8-bit/pixel) images. Figure 5.3 shows that every pixel value of the filtered image is calculated using a corresponding pixel and its eight neighbors in the processed image. The circuit which realizes candidate filters in the FPGA is called *virtual reconfigurable circuit* (VRC).

Virtual reconfigurable circuits were introduced for digital evolvable hardware as a new kind of rapidly reconfigurable platforms utilizing conventional FPGAs [13]. If the VRC configuration is uploaded into the FPGA, there will be created the following units: an array of programmable elements, programmable interconnection network, configuration memory, and configuration port. The VRC is, in fact, a second reconfiguration layer developed on the top of an FPGA in order to obtain fast reconfiguration and application-specific programmable elements. The designer has the opportunity to design the VRC exactly according to requirements of a given application. In most cases the VRC, takes a form of a regular two-dimensional array of programmable elements.

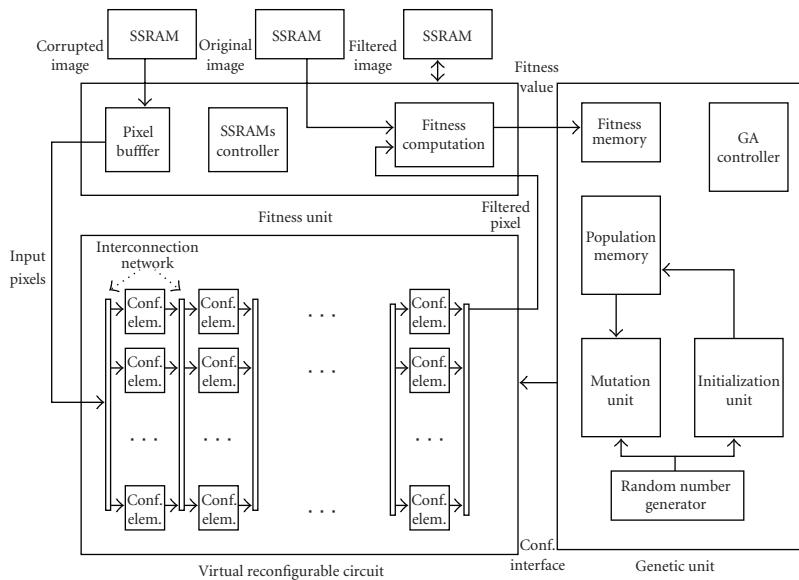
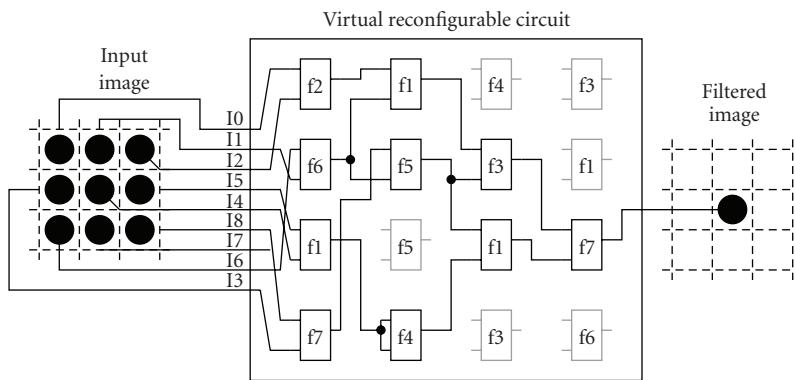


FIGURE 5.2. Architecture of the evolvable image filter in FPGA.

FIGURE 5.3. Example of a 3×3 image operator in VRC.

The VRC can be considered as a hardware implementation of the computational model used in CGP. In contrast to the conventional CGP [10]—where gates and 1 bit connection wires are utilized—configurable functional blocks (CFBs) and 8-bit datapaths are employed here. Our model of the reconfigurable circuit consists of 2-input CFBs placed in a grid of n_c columns and n_r rows (see Figure 5.4). Any input of each CFB may be connected to the primary circuit inputs or to an output of a CFB, which is placed anywhere in the preceding column. The interconnection is implemented using multiplexers. Any CFB can be programmed to realize one of functions given in Table 5.1. These functions were recognized as

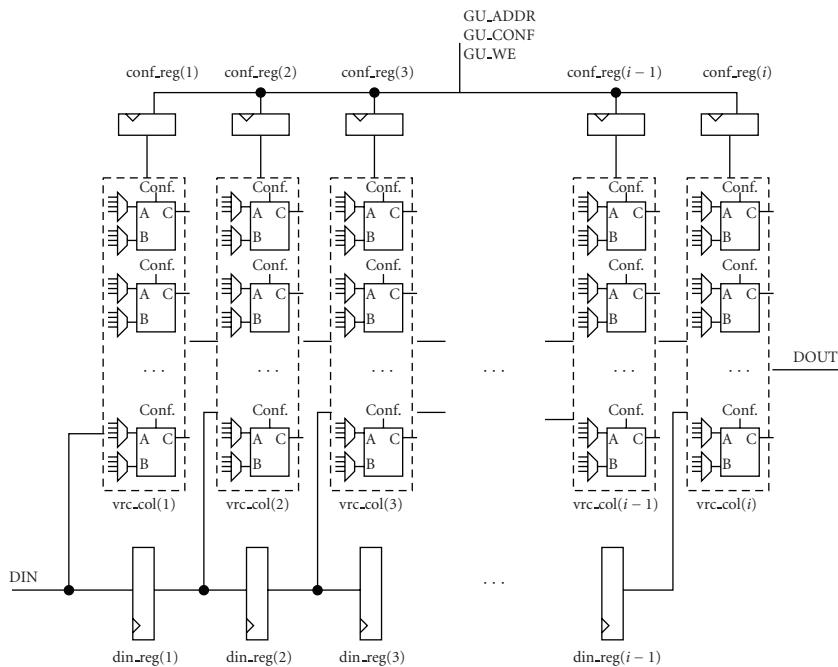


FIGURE 5.4. Virtual reconfigurable circuit for image filtering. Pixels are processed by the array of CFBs and simultaneously stored in the FIFO memory “din_reg” in order to ensure synchronous processing. CFBs are configured using the configuration register “conf_reg” which is controlled by the genetic unit.

useful for this task in [14]. Configuration bits of VRC (stored in a register array) are directly connected to the multiplexers that control the selection of inputs and functions of CFBs. The reconfiguration is performed column by column. The computation is pipelined; a column of CFBs represents a stage of the pipeline. Registers are inserted between the columns in order to synchronize the input pixels with CFB outputs.

Evolutionary algorithm directly operates with configurations of the VRC; simply, a configuration is considered as a chromosome. The chromosome is a sequence of triplets; each of them defines the configuration of one CFB (the connection of its inputs and the function performed by CFB). As 12 bits are used to configure one CFB, the length of chromosome is $12 \cdot n_c \cdot n_r$ bits. The output is always read from the top-right CFB.

It is evident that the VRC implements a combination behavior. As many image filters exhibit this property and pipelining of combinational circuits is very effective, we decided to implement only this type of filters. Of course, a feedback, multiplication operators or time delays could be introduced into the filter in order to obtain more complex behaviors. However, it requires much more resources in FPGA and an additional time to evaluate a candidate circuit. The chosen solution represents a reasonable compromise.

TABLE 5.1. Functions in CFBs operating over two pixels.

	Function	Description
0	$x \vee y$	Binary or
1	$x \wedge y$	Binary and
2	$x \oplus y$	Binary x or
3	$x + y$	Addition
4	$x +^s y$	Addition with saturation
5	$(x + y) \gg 1$	Average
6	$\text{Max}(x, y)$	Maximum
7	$\text{Min}(x, y)$	Minimum

A special interface has been established to read configurations of VRC from FPGA. That makes the analysis of evolved filters easier. Furthermore, any downloaded configuration can be converted to a program (e.g., a function in C language) doing the same filtering task in software.

5.2.2. Fitness calculation in hardware

The fitness calculation is carried out by fitness unit. The pixels of corrupted image u are loaded from an external SRAM memory and then forwarded to inputs of VRC. Pixels of filtered image, v , are sent back to the fitness unit, where they are compared with the pixels of original image, w . Filtered image is simultaneously stored into an additional SRAM memory. Note that all image data are stored in external SRAM memories due to the limited capacity of internal RAMs available in the FPGA chip.

The design objective is to minimize the difference between the filtered image and the original image. The image size is $K \times K$ pixels but only the area of $(K - 2) \times (K - 2)$ pixels is considered because the pixel values at the borders are ignored and thus remain unfiltered. The fitness value of a candidate filter is obtained similarly to Sekanina's original work [12], that is: (1) the VRC is configured using a candidate chromosome, (2) the circuit created is used to produce pixel values in the image v , and (3) the fitness value is calculated as

$$\text{fitness} = \sum_{i=1}^{K-2} \sum_{j=1}^{K-2} |v(i, j) - w(i, j)|. \quad (5.1)$$

The above formula is implemented as a special circuit in FPGA (see Figure 5.5). Similar circuits also ensure reading the original and corrupted images from external memories. This task is not trivial if we consider that nine pixels are needed at each moment of processing and these pixels are not stored at neighboring addresses of the memory. Therefore, special addressing circuits, FIFOs and comparators (used to detect the end of row and the end of picture) are implemented.

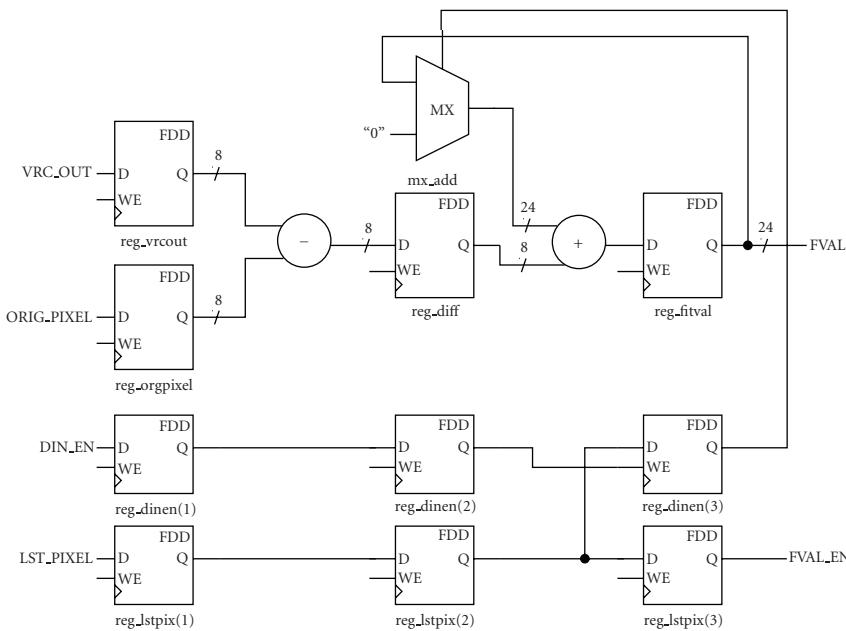


FIGURE 5.5. A circuit calculating the fitness value according to formula (5.1). Every pixel calculated by VRC (VTC_OUT) is compared (symbol “−”) with a corresponding pixel in the reference image (ORIG_PIXEL), and their difference is added (symbol “+”) to the actual fitness value (FVAL). Synchronous processing is assured by registers (FDD).

5.2.3. Hardwired genetic unit

The main advantage of CGP is that the representation used is very natural for description of digital circuits. On the other hand, the representation usually implies a very rugged fitness landscape. Only a simple genetic operator—mutation—is used in the evolutionary algorithm. In contrast to standard genetic programming [7], CGP works with a small population size and many generations are produced (tens of thousands).

All genetic operations are implemented as special circuits in the FPGA. Figure 5.6 shows details of the genetic algorithm datapath. The genetic algorithm we utilized is based only on a single mutation operator (bit inversion). A crossover operator is not taken into account because experiments performed so far have shown that no reasonable crossover operator improves the search. Secondly, its implementation would occupy a considerable area on the FPGA and introduce another delay. The population is stored in a memory and its size is configurable. Another memory is used to store fitness values. Every new population is always generated from the best members of the previous one. Genetic algorithm operates in the following steps.

- (i) Initialization unit generates the first population at random. We use a linear feedback shift register seeded from software. Alternatively, a cellular

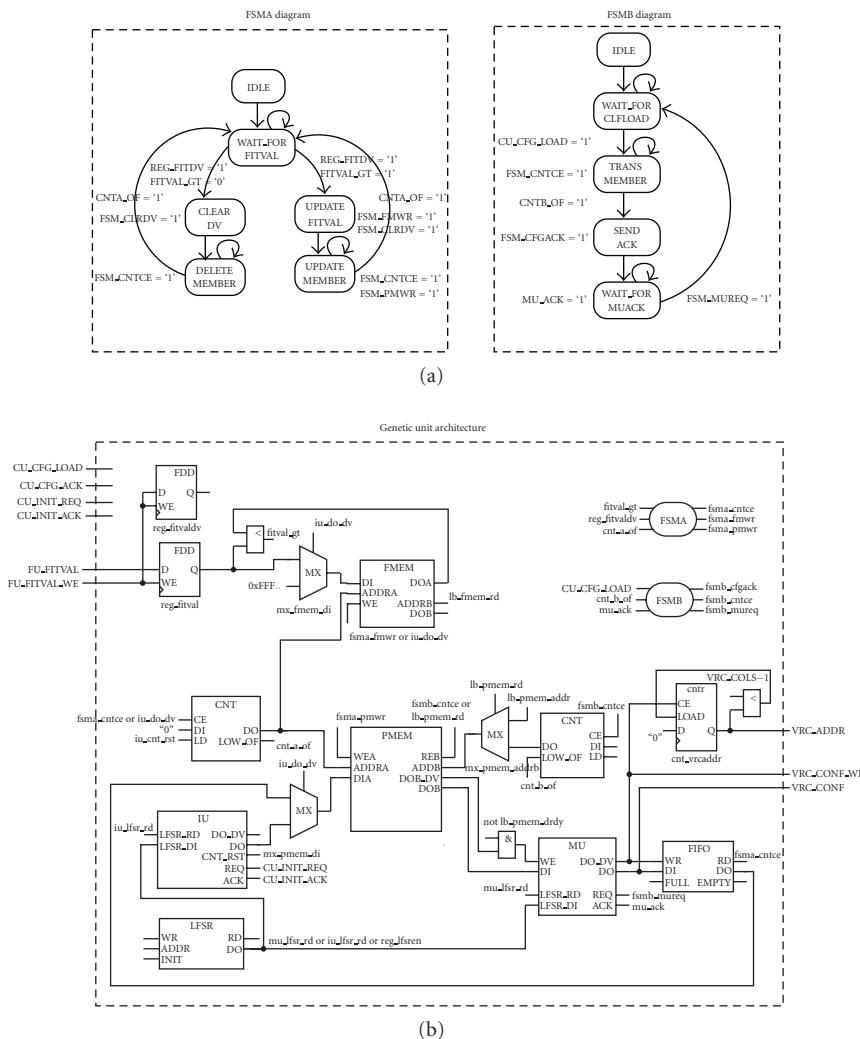


FIGURE 5.6. Block diagram of the genetic unit. Initialization unit (IU) randomly generates the initial population using a linear feedback shift register (LFSR). Two-port population memory (PMEM) stores candidate chromosomes. Its addresses are determined by two counters (CNT). LFSR is also utilized in the mutation unit (MU) to randomly select those genes that will be inverted. The fitness values (FU_{FITVAL}) are stored in fitness memory (FMEM). The genetic unit is controlled using two finite state machines, FSMA, and FSMB. Symbol MX stands for a multiplexer.

automaton could be used. Martin has shown that the performance of these circuits is similar for this task [9].

- (ii) Mutation unit changes a given number of genes (bits) of a population member (this number is configurable) and the modified member is loaded into the VRC; it represents an image operator. The reconfiguration of VRC is pipelined in order to overlap the reconfiguration by

- useful computation (i.e., by filtering the training images by a candidate circuit). Simultaneously, the chromosome is copied into a FIFO memory.
- (iii) Genetic unit is waiting for the evaluation performed by fitness unit. If the fitness value is better than the parent's fitness, then the chromosome replaces its parent (the chromosome is copied from FIFO to the population memory). Otherwise, the chromosome is removed from FIFO.
 - (iv) This is repeated until an appropriate number of generations are produced.

This loop is, in fact, divided into two parts executed concurrently. Because of the partial reconfiguration of VRC, we can send the next population member to VRC although the previous one has not been evaluated yet.

5.2.4. Target platform

Any platform containing a sufficiently large FPGA (approx. 350 k equivalent gates are needed) and external memories can be utilized. We used a COMBO6 card, developed in the Liberouter project [8], which is a PCI card primarily dedicated for a dual-stack (IPv4 and IPv6) router hardware accelerator. This board contains FPGA Virtex XC2V3000 by Xilinx, Inc. with more than 3 million equivalent gates, up to 2 GB DDR SDRAM, up to 9 Mbit context addressable memory, and the three 2 MB SSRAM memories. We decided to use this card for our experiments because it offers us a sufficient performance and capacity of FPGA. Furthermore, various hardware modules (such as a circuit allowing the communication with a personal computer via PCI bus) and some design software are available for free.

5.2.5. Results of synthesis

The evolvable image filter was described in VHDL, simulated using ModelSim and synthesized using LeonardoSpectrum and Xilinx ISE tools to Virtex FPGA XC2V3000bf957 chip which is available on the COMBO6 card. In order to compare different implementations, we have decided to synthesize the whole system with VRC of size 4×8 , 5×8 , 6×8 , and 7×8 CFBs. The evolutionary algorithm operates in the same way for all implementations; however, the size of chromosome depends on the number of CFBs. Table 5.2 shows the resources increase in FPGA with the increasing number of CFBs.

5.2.6. Time of evolution

The evaluation of a candidate filter consists of three basic activities: (1) preparation of a new candidate chromosome (filter), (2) reconfiguration of VRC circuit according to the prepared chromosome, and (3) fitness evaluation of the filter. As most time is spent in the fitness evaluation, the architecture of evolvable image filter is designed in order to overlap the fitness evaluation by other activities (1, 2).

TABLE 5.2. Results of synthesis for Xilinx Virtex II xc2v3000 FPGA. The table shows the utilization of function generators, registers (Dffs) and latches, block RAMs, and input–output blocks of the FPGA for various VRC sizes.

VRC size available	Func. gens.	Dffs or latches	Block RAMs	IO Blocks
	28.672	28.672	96	684
4 × 8 utilization	10.331 36%	3.104 10%	2 2%	236 34%
5 × 8 utilization	12.324 42%	3.269 11%	2 2%	236 34%
6 × 8 utilization	15.861 55%	3.468 12%	2 2%	236 34%
7 × 8 utilization	17.753 61%	3.632 12%	3 3%	236 34%

Therefore, because there is no overhead for reconfiguration of VRC (a new candidate configuration is prepared as well as VRC is reconfigured during evaluation of a previous filter) it is possible to express the time of evaluation of a single filter as

$$t_{\text{eval}} = (K - 2)^2 \cdot \frac{1}{f} = (256 - 2)^2 \cdot \frac{1}{50 \cdot 10^6} = 1.29 \text{ ms} \quad (5.2)$$

if the size of images is 256×256 pixels and the FPGA operates at 50 MHz. Time of evolution can be expressed as follows:

$$t_e = t_{\text{init}} + g \cdot n \cdot t_{\text{eval}}, \quad (5.3)$$

where g is the number of generations, n is population size and t_{init} is time needed to generate the initial population (t_{init} is negligible). Considering $n = 4$ and $g = 10,000$, the time of evolution is 51.6 seconds for the 256×256 pixel images with FPGA running at 50 MHz.

5.3. Experimental evaluation of the FPGA implementation

Because of the stochastic nature of the method, it is impossible to evaluate the proposed architecture for all possible types of images, noise, time of evolution, and setting of CGP parameters. The following experiments were arranged to evaluate the most important properties of the evolvable unit, such as time of evolution and the quality of evolved filters. In particular we investigated the following:

- (i) the influence of the mutation rate on the time of evolution;
- (ii) the influence of the size of VRC on the quality of evolved filters;

- (iii) the influence of the population size;
- (iv) the quality of evolved filters for different size of training images;
- (v) the quality of evolved filters for different types of noise and images;

The following problems were utilized as benchmark tasks for the proposed architecture (see examples in Figure 5.10):

- (i) noise removal from images corrupted by Gaussian noise ($\sigma = 16$);
- (ii) noise removal from images corrupted by salt and pepper noise (5% pixels with white or black shots);
- (iii) noise removal from images corrupted by random shot noise (5% pixels with a random value of shot);
- (iv) edge detection.

These benchmark problems were utilized in literature dealing with the evolutionary design of image filters [13–15, 20]. Conventional realizations of these filters are based on mean and median filtering. There are many conventional edge detectors, for example, Sobel and Canny detectors. References [13, 16] discuss these conventional filters and their implementation cost in hardware.

The choice of training image depends on a given application. The proposed implementation supports up to 256×256 -pixel images. We have utilized the standard Lena image in our experiments. The resulting filters are verified using a test set consisting of various images. The quality of filtered images is expressed in terms of *mean difference per pixel* (mdpp). This value is obtained by dividing the fitness value (5.1) by the number of filtered pixels. Alternatively, signal-to-noise ratio (snr) is used in literature; however, it is easier to implement a circuit calculating mdpp than snr in FPGA.

The genetic unit attempts to minimize mdpp. However, a zero value of mdpp is unreachable even for a very long time of evolution. The reason is that considered types of noise contain random components which cannot completely be removed by the filters that can be implemented in the VRC.

5.3.1. The mutation rate versus the quality of resulting filters

In our case, only experimental work can suggest the suitable mutation rate. We repeated the filter evolution 100 times for the salt and pepper noise problem using the 128×128 pixel Lena image, population size 8, and 4×8 CFBs in VRC. Minimum, maximum, and average fitness values were measured after 50000 generations. Figure 5.7 shows that it is useful to mutate 10 bits (2.5%) of the chromosome in average.

In order to improve the quality of the search method we utilized a simple adaptive mutation scheme. In case that the best fitness value stagnates for a certain number of generations (g_c), a higher mutation rate is introduced in order to escape from the local optimum. As soon as the fitness value is improved, the mutation ratio is decreased back to the previous value. We experimentally investigated various values of g_c and mutation rates. However, no improvement against the constant mutation rate was observed after more than 500 runs of CGP with

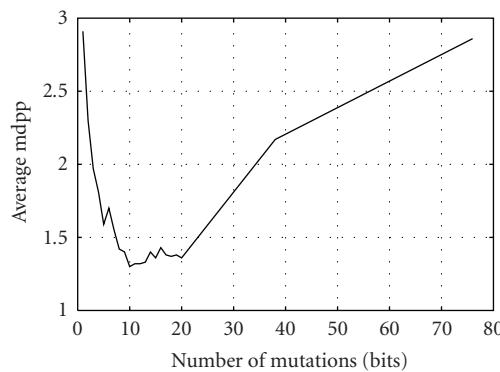


FIGURE 5.7. Searching for a suitable mutation rate (chromosome length is 384 bits).

TABLE 5.3. The quality of filters evolved for various VRCs.

VRC size	Best mdpp	Average mdpp	Average number of gener.
4×8	0.50	1.92	28911
5×8	0.38	1.72	27310
6×8	0.35	2.21	27025
7×8	0.40	2.23	27404

various parameters. Therefore, the adaptive mutation was not considered in the final implementation.

5.3.2. The size of VRC versus the quality of evolved filters

Similarly to the probability of mutation, the size of VRC also influences the quality of the evolved filters and time of evolution. We repeated the filter evolution 500 times for the salt and pepper noise problem using the 128×128 pixel Lena image, population size 4, mutation rate 10 bits/chromosome and VRC consisting of 4×8 , 5×8 , 6×8 and 7×8 CFBs. All experiments were terminated after 50000 generations. The best mdpp, average mdpp, and the average number of generations in which the evolution has stagnated are reported in Table 5.3. The results indicate that, in average, the best mdpp is obtained for 5×8 CFBs.

5.3.3. Population size

In this experiment we allowed performing 160000 fitness evaluations and measured whether it is better to maintain larger population or to produce more generations. For the salt and pepper noise problem we repeated the filter evolution 200 times for each setup using the 128×128 pixel Lena image, mutation rate 10 bits/chromosome, and with 4×8 CFBs in VRC. The best mdpp and average mdpp are reported in Table 5.4.

TABLE 5.4. The number of generations versus population size for 160000 evaluations.

Population size	Generations	Best mdpp	Average mdpp
4	40 k	0.42	1.89
8	20 k	0.53	1.61
16	10 k	0.59	1.56
32	5 k	0.70	1.51

TABLE 5.5. The best filters evolved for different sizes of training images.

Image	Size	Best mdpp	Avr. mdpp	Filter
Lena256.bmp	256 × 256	0.31	0.92	F256b
Lena128.bmp	128 × 128	0.40	1.34	F128b
Lena64.bmp	64 × 64	0.38	1.53	F64b
Lena32.bmp	32 × 32	0.50	1.89	F32b

In our implementation, large populations imply lower mdpps in average; however, it is difficult to obtain at least one very good solution. Thus, we can suggest that if one is going to use the evolvable filter in FPGA to find a really good filter, it is better to generate more generations with a small population size. On the other hand, when the system should be used to achieve “real-time” adaptation, it is better to maintain larger population because the probability that the resulting solution is good is higher (assuming the same time available to the evolutionary design in both cases).

5.3.4. The quality and generality of filters evolved for different size of training image

Another question is how many pixels the training image has to contain. Naturally, larger training images should lead to more general filters; on the other hand, many pixels have to be examined in the fitness function which makes the evolution very time consuming. In comparison to the 256 × 256 pixel training image, the evolutionary process will be 4 times faster if only 128 × 128 pixels are examined, 16 times faster for 64 × 64 pixels, and 64 times faster for 32 × 32 pixels. We evaluated the mentioned scenarios on the salt and pepper noise removal problem for Lena image. Figure 5.8 shows the images utilized in the fitness function. We repeated the filter evolution 500 times (50000 generations in each run) for each size of training image, population size 10, mutation rate 8 bits/chromosome, and 4 × 8 CFBs in VRC. We applied the best four evolved filters (F256b, F128b, F64b, F32b) on other two images and calculated their mdpp (see Tables 5.5 and 5.6).

It seems that the use of the 128 × 128 pixel training image is a reasonable compromise. The usage of smaller training images leads to filters that are not general (i.e., they are optimized only for the training image). However, we can imagine a real-world application in which it could be useful to have a filter optimized not

TABLE 5.6. The application of evolved filters on different images (mdpps are given).

Image/filter	F256b	F128b	F64b	F32b
Lena256p5.bmp	0.31	0.48	0.55	0.76
Bird256p5.bmp	0.27	0.24	0.29	0.33
Bridge256p5.bmp	0.90	0.80	1.17	0.95
Average	0.49	0.51	0.67	0.68

FIGURE 5.8. Training images utilized in the fitness function: 256×256 , 128×128 , 64×64 , and 32×32 pixels.

only for a given type of noise but also for certain image(s). For example, a camera situated over a production line (say, with bottles) scans very similar types of images (bottle by bottle). The filter evolved using a small training image would be an advantage because of short adaptation time (a few seconds).

5.3.5. The quality of filters evolved for different types of noise

Table 5.7 summarizes the results we obtained in searching for the best filter for Gaussian noise, salt and pepper noise, random shot noise, and edge detector. We repeated the evolution 500 times (50000 generations in each run) for all noise types using the 256×256 pixel Lena image, population size 4, mutation rate 10 bits/chromosome, and 4×8 CFBs in VRC. The third column of Table 5.7 indicates the generation in which the best filter was discovered. The last column represents the average number of generations in which the evolution has stagnated. The best mdpp will be compared against the software approach in Section 5.4. We applied the evolved filters on other test images. Figures included in Figure 5.9 demonstrate that the visual quality of the filtered images is sufficient and the evolved filters work correctly for a considered class of images. All types of noise and examples of filtered images are visible in Figure 5.10. Example of a filter evolved to suppress Gaussian noise is depicted in Figure 5.11.

5.4. Discussion

Every day, during our experimental work, we were able to evolve more than 4500 image filters (considering a training image of 128×128 pixels, population size 4,

TABLE 5.7. The best filters evolved for test problems.

Problem	Best mdpp	Number of gen.	Avr. mdpp	Std.dev.	Avr. Number of gen.
Salt & pepper	0.31	49808	0.95	0.51	29388
Random shot	1.11	40616	1.65	0.31	26781
Gaussian noise	6.36	27179	6.73	0.24	31032
Edge detection	1.16	49608	1.73	0.41	35074

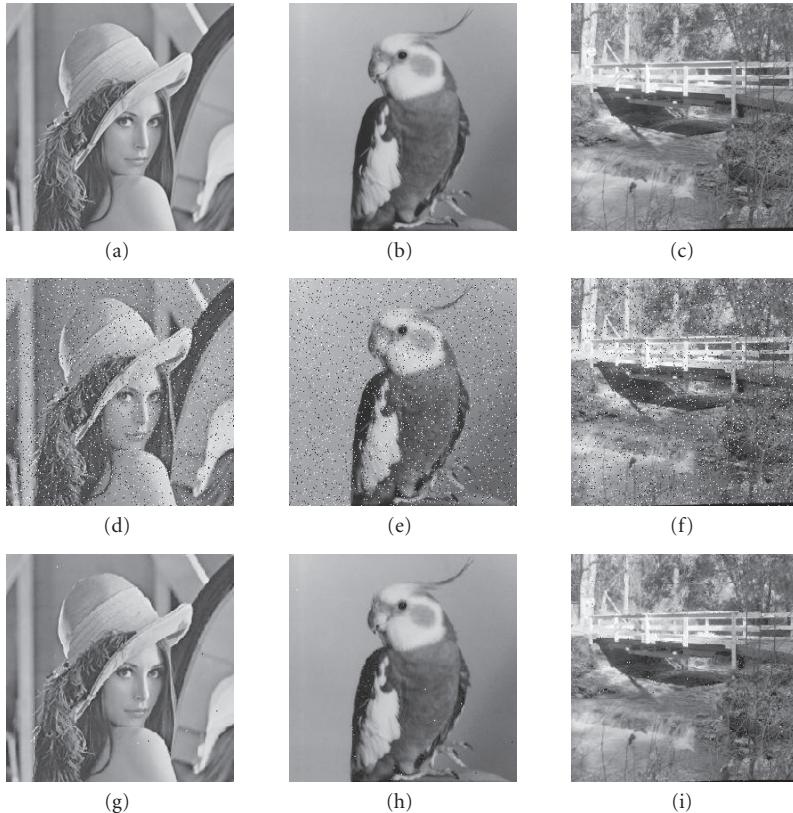


FIGURE 5.9. Original images (a), (b), (c), images with the salt and pepper noise from a test set (d), (e), (f), and images filtered using an evolved filter (g), (h), (i). The evolved filter was trained by means of Lena image (a), (d).

and 30000 generations in average) which is impossible by means of a PC-based software approach. As the hardwired evolutionary design process is very fast, we could explore much larger portion of the design space than by using a conventional approach. The change of problem specification is very easy; the user has to supply a corrupted image and an uncorrupted image (these images are uploaded into RAMs at COMBO6) and some parameters of evolution (the mutation probability



FIGURE 5.10. Corrupted and filtered images: salt and pepper noise ((a) versus (e)), random shot noise ((b) versus (f)), Gaussian noise ((c) versus (g)), edge detector ((d) versus (h)).

and the number of generations). In case that more changes are required (e.g., VRC is changed substantially), it is necessary to perform a new synthesis of the whole application which requires a few minutes.

From Table 5.7 it can be derived that 30000 generations (i.e., 20 seconds in FPGA operating at 100 MHz) are needed in average to find a filter. The design time is very reasonable if the proposed system should operate “instead” of a designer in the image filter design task. For some applications, our solution could also operate as a real-time evolving filter. If we consider that training images could consist only of 64×64 pixels, then the time of evolution is 4.7 seconds. Note that the speedup we obtained against the software approach (Pentium III/800 MHz) is 50 if the FPGA operates at 100 MHz. Another speedup is possible if more than one VRC were implemented.

Table 5.8 compares the best-achieved results with the results obtained using a software simulator [14] and with the conventional implementations. We can observe that the filters evolved here and in [14] exhibit very similar quality. In comparison with conventional implementations of image filters (i.e., with the median or mean filters that are general and not specialized to our type of noise) we obtained very good specific filters trained for a particular type of noise. Similar to [14], we can assume that the evolved filters, when extracted from VRC, transformed to VHDL, and synthesized again for FPGA, will be cheaper than conventional filters (in terms of the number of equivalent gates utilized in FPGA).

The performed experiments suggest how the final architecture of the evolvable image filter should look like. The circuit should contain 5×8 CFBs, CGP should mutate 2.5% of chromosome and the fitness value should be calculated using the 128×128 pixel image in order to evolve general filters. If the system is utilized to design a single filter, then it is useful to maintain a small population size (four

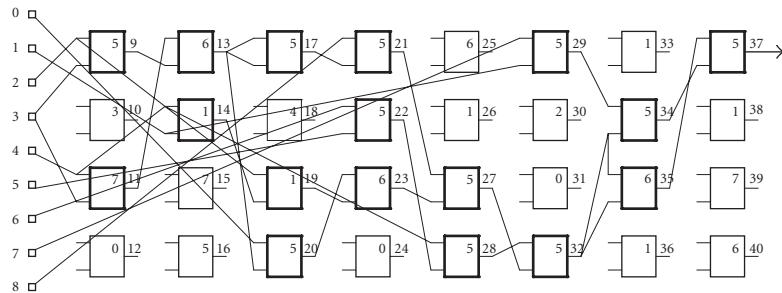


FIGURE 5.11. Example of an evolved Gaussian noise filter. CFBs are numbered according to Table 5.1.

TABLE 5.8. Comparison of mdpp of the best filters evolved in software (cf. [14]), in hardware and designed conventionally (cf. [14]).

Operator	SW	HW	Conventional
Gaussian	6.24	6.36	6.43 (mean)
Salt and pepper	0.38	0.31	2.95 (median)
Random Shot	1.08	1.11	2.98 (median)
Edge detection	1.20	1.16	—

chromosomes). On the other hand, when a short adaptation time is of interest, larger populations allow obtaining higher average quality of filters.

A strongly generic approach was utilized during VHDL design. All the implemented units are parameterized using various constants (such as the size of chromosome and the number of mutations). Therefore, it is easy to modify the design and to obtain a totally different evolvable system in a very short time. The FPGA communicates with PC via a special software allowing the designer to prepare scripts describing experiments that have to be performed. Typically, designer specifies the VRC, EA, and fitness functions, performs synthesis, uploads the evolvable system into FPGA, and executes all experiments described in scripts. This approach can be considered as a user-friendly interface to evolvable hardware (e.g., a system for hardware evolution of sorting networks was derived from this application in a few hours). Furthermore, the evolvable image filter can be offered as an IP (intellectual property) core to FPGA designers. Another important feature is that the evolvable filter can be uploaded to the FPGA dynamically, that is, only when some adaptation is needed. Once the system is adapted, only a resulting filter remains in the FPGA (e.g., in a second VRC realized in the FPGA) and some other circuits can replace the evolvable unit in the FPGA.

5.5. Conclusions

The proposed complete hardware implementation of an evolvable image filter has allowed us to explore much larger portion of the design space than engineers can do nowadays (e.g., during an eight-hour work period). The proposed system is

able to deliver image filters of a high quality in a few minutes. In case that the evolved filter is not sufficient, it is needed just to restart the evolutionary process. The architecture is generic and can easily be modified to realize other evolvable systems. We demonstrated that complete hardware evolution significantly improves designer's performance and allows integrating novel features (such as adaptability) to FPGA-based embedded systems.

Acknowledgments

This research was performed with the Grant Agency of the Czech Republic under contract no. 102/07/0850 *Design and hardware implementation of a patent-invention machine* and the Research intention no. MSM 0021630528-Security-Oriented Research in Information Technology.

Bibliography

- [1] A. Burian and J. Takala, "Evolved gate arrays for image restoration," in *Proceedings of the Congress on Evolutionary Computation (CEC '04)*, vol. 1, pp. 1185–1192, Portland, Ore, USA, June 2004.
- [2] A. deHon, "Comparing computing machines," in *Configurable Computing: Technology and Applications*, vol. 3526 of *Proceedings of SPIE*, pp. 124–133, Bellingham, Wash, USA, November 1998.
- [3] J. Dumoulin, J. A. Foster, J. F. Frenzel, and S. McGrew, "Special purpose image convolution with evolvable hardware," in *Real-World Applications of Evolutionary Computing—Proceedings of the 2nd Workshop on Evolutionary Computation in Image Analysis and Signal Processing (EvoIASP '00)*, vol. 1803 of *Lecture Notes in Computer Science*, pp. 1–11, Springer, Edinburgh, UK, April 2000.
- [4] M. Erba, R. Rossi, V. Liberali, and A. Tettamanzi, "An evolutionary approach to automatic generation of VHDL code for low-power digital filters," in *Proceedings of the 4th European Conference on Genetic Programming (EuroGP '01)*, vol. 2038 of *Lecture Notes in Computer Science*, pp. 36–50, Springer, Milan, Italy, April 2001.
- [5] T. Higuchi, T. Niwa, T. Tanaka, H. Iba, H. de Garis, and T. Furuya, "Evolving hardware with genetic learning: a first step towards building a Darwin machine," in *Proceedings of the 2nd International Conference on Simulated Adaptive Behaviour*, pp. 417–424, MIT Press, Honolulu, Hawaii, USA, 1993.
- [6] G. Hollingworth, A. Tyrrell, and S. Smith, "Simulation of evolvable hardware to solve low level image processing tasks," in *Proceedings of the Evolutionary Image Analysis, Signal Processing and Telecommunications Workshop (EvoWorkshops '99)*, vol. 1596 of *Lecture Notes in Computer Science*, pp. 46–58, Springer, Göteborg, Sweden, May 1999.
- [7] J. R. Koza, F. H. Bennett III, D. Andre, and M. A. Keane, *Genetic Programming III: Darwinian Invention and Problem Solving*, Morgan Kaufmann, San Francisco, Calif, USA, 1999.
- [8] "Liberouter project," <http://www.liberouter.org/>, 2005.
- [9] P. Martin, *Genetic programming in hardware*, Ph.D. thesis, University of Essex, Colchester, UK, 2003.
- [10] J. F. Miller, D. Job, and V. K. Vassilev, "Principles in the evolutionary design of digital circuits—part I," *Genetic Programming and Evolvable Machines*, vol. 1, no. 1-2, pp. 7–35, 2000.
- [11] P. Porter, *Evolution on FPGAs for feature extraction*, Ph.D. thesis, Queensland University of Technology, Brisbane, Australia, 2001.
- [12] L. Sekanina, "Image filter design with evolvable hardware," in *Applications of Evolutionary Computing—Proceedings of the 4th Workshop on Evolutionary Computation in Image Analysis and Signal Processing (EvoIASP '02)*, vol. 2279 of *Lecture Notes in Computer Science*, pp. 255–266, Springer, Kinsale, Ireland, April 2002.

- [13] L. Sekanina, *Evolvable Components: From Theory to Hardware Implementations*, Natural Computing, Springer, Berlin, Germany, 2004.
- [14] L. Sekanina and R. Ruzicka, "Easily testable image operators: the class of circuits where evolution beats engineers," in *Proceedings of the NASA/DoD Conference on Evolvable Hardware*, pp. 135–144, IEEE Computer Society, Chicago, Ill, USA, July 2003.
- [15] S. L. Smith, S. Leggett, and A. M. Tyrrell, "An implicit context representation for evolving image processing filters," in *Applications of Evolutionary Computing*, vol. 3449 of *Lecture Notes in Computer Science*, pp. 407–416, Lausanne, Switzerland, March–April 2005.
- [16] M. Sonka, V. Hlaváč, and R. Boyle, *Image Processing: Analysis and Machine Vision*, Thomson-Engineering, Toronto, Canada, 1999.
- [17] G. Tufte and P. C. Haddow, "Prototyping a GA pipeline for complete hardware evolution," in *Proceedings of the 1st NASA/DoD Workshop on Evolvable Hardware*, A. Stoica, D. Keymeulen, and J. Lohn, Eds., pp. 18–25, IEEE Computer Society, Pasadena, Calif, USA, July 1999.
- [18] "Xilinx Inc," <http://www.xilinx.com/>, 2005.
- [19] Y. Zhang, S. L. Smith, and A. M. Tyrrell, "Digital circuit design using intrinsic evolvable hardware," in *Proceedings of the NASA/DoD Conference on Evolvable Hardware*, pp. 55–62, Seattle, Wash, USA, June 2004.
- [20] Y. Zhang, S. L. Smith, and A. M. Tyrrell, "Intrinsic evolvable hardware in digital filter design," in *Applications of Evolutionary Computing*, vol. 3005 of *Lecture Notes in Computer Science*, pp. 389–398, Springer, Coimbra, Portugal, April 2004.

Lukáš Sekanina, Faculty of Information Technology, Brno University of Technology, Božetěchova 2, 612 66 Brno, Czech Republic

Email: sekanina@fit.vutbr.cz

Tomáš Martínek, Faculty of Information Technology, Brno University of Technology, Božetěchova 2, 612 66 Brno, Czech Republic

Email: martinto@fit.vutbr.cz

6

Variable-length compositional genetic algorithms for the efficient implementation of morphological filters in an embedded image processor

Ian P. W. Sillitoe and Andreas K. Magnusson

6.1. Introduction

High-speed and ultra-high-speed binary image processing has a growing number of application areas in manufacturing industry and transportation (e.g., vision-based fabric inspection [19], intelligent vehicles [2]). The relative simplicity of the hardware required for high-speed binary image processing can also be used to provide a cost effective means of implementing grey scale image processing in mobile embedded systems [3].

A common component of such embedded vision systems is the use of morphological binary image processing. This is dictated by real-time demands and the requirement that the application must often measure or classify the morphological properties of the image. In applications where the image sizes are large, the image processing operators are implemented by passing the entire image, line by line, through a scan-line pipeline [6]. Thus, the time-complexity of an algorithm is typically $O(np)$, where n is the number of lines in the image and p is the number of individual operator applications. Whilst it is not possible to alter n , it is possible to reduce the execution time of an algorithm by combining and reordering the original structuring element operator sequence into a shorter sequence of multiminterm¹ Boolean operators. Shorter sequences not only provide the benefit of shorter execution times but they can also provide a corresponding power saving when used in conjunction with variable frequency clock modes.

Thus, the aim of the optimisation procedure is to map the original filter specification into a reduced sequence of machine specific operators and connectives. The optimisation criterion is specified by the length of the sequence, and the

¹Boolean functions, such as a sequence of structuring elements, can be represented canonically as the disjunction of a series of minterms. A minterm consists of a conjunction of each of the function variables (either complemented, or uncomplemented) where each function variable appears exactly once within a given minterm (e.g., the exclusive or function would be represented as two minterms X or $(a, b) = (\bar{a} \wedge b) \vee (a \wedge \bar{b})$).

complexity of the optimisation problem is determined by the size of the operator set and the possible number of partitions of the target filter.

The genetic algorithm, presented here, evolves compact sequences of multi-minterm operators for the *clutter* [16], a high-speed scalable FPGA-based binary image processor. The *clutter* has a large and complex multiminterm instruction set based upon the functions realisable by the compact look-up table (CLUT) [17]. The size of the instruction set is orders of magnitude larger than those of previous processors (an upper bound of 2^{76} realisable functions [17]) and it is impractical to enumerate the entire instruction set. The size and form of the instruction set provide the prerequisites for the optimisation of complex filters (such as the median or rank filters [18]) lacking in other architectures [9].

The number of disjoint subset partitions of target function minterms is related to the Bell number, B_n , given in

$$B_n = \sum_{k=1}^n S_k^{(n)}, \quad (6.1)$$

where n is the number of target function minterms and $S_k^{(n)}$ are the Stirling numbers of the second kind, calculated by the recurrence relation in

$$S_k^{(n)} = S_{k-1}^{(n-1)} + k \cdot S_k^{(n-1)}, \quad S_1^{(n)} = S_n^{(n)} = 1. \quad (6.2)$$

For a target function containing 16 minterms, the number of possible disjoint minterm partitions is $B_{16} = 10480142147$. The Bell number becomes a lower bound on the number of partitions, when applied to problems involving overlapping subsets and arbitrary connectives (as is the case here). Thus, even for small numbers of minterms, the combination of the partition space and the size of the instruction set makes the use of simple deterministic backtracking algorithms impractical.

In previous work, an informed backtracking algorithm based upon the results of [17] was successful in identifying solutions to target filters which could be realised by a single CLUT. This approach was extended, to tackle more complex filter functions, by recursively applying the algorithm to the residue of each solution. However, the solutions discovered by the extended approach were found to be similar in length to those of the original structuring element sequences.

6.2. Synthesis strategy

The informed backtracking algorithm partitions the target function into a disjunctive sequence of single CLUT operators. Thus, only those minterms contained within the target function are implemented by the CLUT operators, and each CLUT operator can only add to the total set of minterms. The disjunctive nature of this form of sequence provides only limited possibilities for optimisation, and does not exploit the richness of the instruction set.

TABLE 6.1. List of the arbitrary connectives.

Function	Encoding	Function	Encoding
$a \wedge b$	0001	$a \vee \bar{b}$	1000
$a \wedge \bar{b}$	0010	$a \vee \bar{b}$	1001
$\bar{a} \wedge b$	0100	$a \vee \bar{b}$	1011
$a \leq b$	0110	$\bar{a} \vee b$	1101
$a \vee b$	0111	$a \wedge \bar{b}$	1110

A more effective strategy is to synthesise sequences in which the CLUT operators cover minterms other than those found within the target function. Successive CLUT operators in the sequence could then remove the unnecessary minterms with the aid of a more complex set of connectives. Such a strategy has the potential to construct shorter solutions based upon a number of greedy CLUT operators which utilises the entire instruction set.

The synthetic strategy can be implemented in a number of ways. The particular form of CLUT operator sequence, S , investigated here, is given in

$$S(X) = (\dots ((\emptyset \circ_1 C_1(X)) \circ_2 C_2(X)) \dots) \circ_N C_N(X), \quad (6.3)$$

where $C_1(X) \dots C_N(X)$ are the CLUT operators applied to the image X , and $\circ_1 \dots \circ_N$ are any of the connectives given in Table 6.1. C_1 is combined with a constant image \emptyset using the connective \circ_1 . Thus, the operators in (6.3) are ordered and, in general, a CLUT operator can completely or partially redefine the effects of the preceding operators.

The advantage of the formulation given in (6.3) is twofold. Firstly, the fitness evaluation of the genetic algorithm, when using (6.3), is only based upon the contents of the CLUT operator, connectives, and the truth table of the target function and is not dependent upon the image. Secondly, the formulation can also be implemented as parallel CLUT operations applied to a single input image.

Section 6.3 provides a brief introduction to the *clutter* and to the CLUT, from which the problem derives a significant portion of its complexity. Section 6.4 introduces the general approach taken by the algorithm to manage the synthesis of solutions. Section 6.5 describes the details of the genetic algorithm, including the diversity management and adaptive niche focusing scheme. Section 6.6 presents the results of the algorithm when applied to complex filter problems. Section 6.7 provides an analysis of the results and observations for further work.

6.3. The clutter

The *clutter* is a novel reconfigurable FPGA-based binary image processor, which is designed to address the requirements of embedded machine vision applications (i.e., cost, power consumption, and processing rate). The architecture addresses

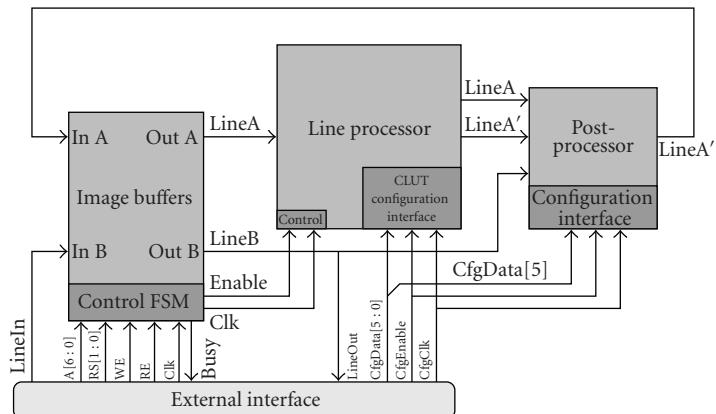


FIGURE 6.1. A block diagram of the core of the *clutter* architecture. Signals pertaining to the inter-*clutter* interface have been intentionally excluded for clarity.

these requirements through its augmentation of the traditional scan-line architecture and the employment of the compact look-up table as the means of neighbourhood transformation. The use of the CLUT conveys, not only, the benefit of a large instruction set but also provides a dense mapping of the *Clutter* architecture to cellular logic-based FPGAs.

6.3.1. The clutter architecture

The *clutter* reference architecture (illustrated in Figure 6.1) consists of three logical entities—the line processor, the postprocessor, and image buffers. The current version of the *clutter* is able to process over 150,000 frames of 256×256 pixels a second, and is implemented within a single low-cost FPGA.

The *clutter* implements image transformations by passing consecutive lines of an image, contained within an image buffer, through the line processor. The contents of the CLUTs define the form of transformation to be applied. The postprocessor combines the transformed lines, according to the connective chosen from Table 6.1, with the corresponding lines in another image buffer. The final result is then written back, a line at a time, to a selected image buffer. Thus, the time taken to process an image is proportional to the number of transform-connective pairs contained within the sequence, given in (6.3).

6.3.2. The compact look-up table

The image transformation realised by a CLUT operator is a general form of the hit-or-miss transformation on a 3×3 neighbourhood. The CLUT is a two-layer structure consisting of five 4-input binary look-up tables, where certain of the inputs are shared between the first layer of the structure (see Figure 6.2(a)). The

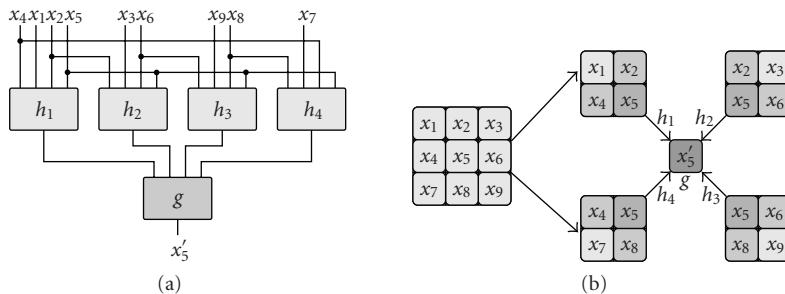


FIGURE 6.2. (a) The internal structure of the compact look-up table (CLUT). The configuration signals for the look-up tables have been excluded for clarity. (b) The form of CLUT neighbourhood mapping. The 3×3 neighbourhood is grouped into four partially overlapping segments, which are processed individually by the first layer look-up tables. The results from the four segments are merged into the transformed centre pixel, x'_5 , by the function implemented by the look-up table g .

single look-up table, g , in the second layer generates an output which is an arbitrary function of the outputs of the first layer. The pixel neighbourhood mapping which results is shown in Figure 6.2(b).

The functions, $f(\cdot)$, realised by the CLUT are a subset of the set of all 2^{512} Boolean functions of nine variables, and have the form

$$f(x_1, \dots, x_9) = g(h_1(\cdot), h_2(\cdot), h_3(\cdot), h_4(\cdot)), \quad (6.4)$$

where $h_1 \dots h_4$ and g implement arbitrary Boolean functions

$$\left. \begin{array}{l} h_1(x_4, x_1, x_2, x_5) \\ h_2(x_2, x_3, x_6, x_5) \\ h_3(x_6, x_9, x_8, x_5) \\ h_4(x_8, x_7, x_4, x_5) \\ g(h_1, h_2, h_3, h_4) \end{array} \right\} : \{0, 1\}^4 \longrightarrow \{0, 1\} \quad (6.5)$$

and x_1, \dots, x_9 are the Boolean values of the 3×3 image neighbourhood (shown in Figure 6.2).

Figure 6.3 illustrates the sensitivity of the CLUT for internal single bit transitions. The sensitivity is measured as the distribution of Hamming distances from the nominal function (generated by a CLUT with randomly generated contents), when a single bit is perturbed at a specific position within the CLUT. It can be seen that transitions in the contents of the second layer look-up table, g , cause a larger divergence in the implemented function than those of the first layer. The changes which occur in the function are a result of the complex one-to-many mapping between the inputs and the output. The form of the mapping makes it difficult to affect isolated minterms within the function.

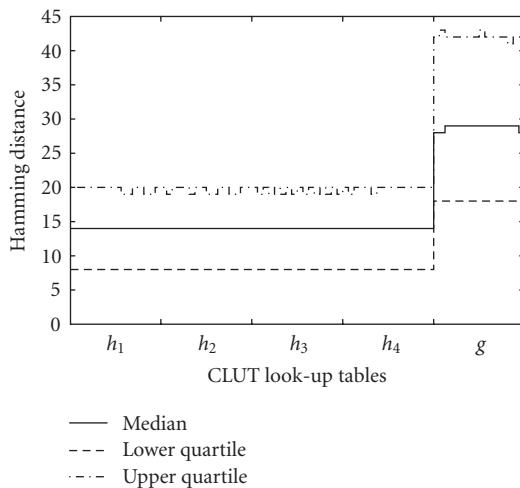


FIGURE 6.3. A graph illustrating the sensitivity of the CLUT structure measured as deviations from the nominal function to isolated bit transitions in the look-up table structure.

6.4. Managing synthesis

The synthesis of sequences (such as in (6.3)) requires that a genetic algorithm provide evolutionary operators which modify the length of individuals and a means of maintaining diversity within a population of variable length individuals. The following section discusses the choices for this particular algorithm in terms of characteristics of the problem.

6.4.1. Variable-length genomes

Compositional evolutionary operators, such as crossover, combine large pieces of preadapted genetic material, and rely upon the building-block hypothesis to yield fitter individuals [8]. The sources of the preadapted material can either be derived from the same individual or from one or more individuals in the population. The motivation for the inclusion of constructive compositional operators within the algorithm in order to grow solutions is twofold.

Firstly, the determination of the optimal length of the final solution is the objective of the evolutionary search, and since no information is available to inform the initialisation process, a trial and error procedure would be required in order to determine a range of suitable lengths. Thus, this application differs significantly from other variable-length applications (e.g., [1]) where a priori information is available to initialise the population.

Secondly, even if a suitable range of lengths were known, the population must be initialised in such a way, that is neither deleterious nor biased. Owing to the characteristics of the fitness landscape, the random initialisation of a population

of variable length individuals results in longer individuals with low initial values of fitness, and shorter individuals with relatively high values of initial fitness. The effect upon the evolutionary dynamics is that small improvements in fitness of a few short individuals rapidly reduces the genetic diversity of the population, and leads to premature convergence or failure to evolve a solution.

If, on the other hand, the population is initialised with individuals containing single CLUT operators and these individuals are grown using constructive operators (such as concatenation), the initial population will have a higher fitness diversity, and this diversity consequently decreases less after the initial settling period (owing to greater variation in the population's ancestors [4]). Experiments illustrating this effect are presented in Section 6.6.

6.4.2. Niche measures in diversity management schemes

Diversity management methods are introduced into genetic algorithms in order to ameliorate the problem of premature convergence in the face of complex fitness landscapes. These methods effectively partition the population into niches, and ensure that competition between individuals is restricted to those which occupy the same niche. The existence of niches allow multiple regions of the search space to be explored simultaneously. Several such methods have been proposed [15].

Thus, a modified diversity management scheme could be designed to niche groups of individuals within the population according to the length of the sequence, and thereby mitigate the problem identified in the previous section. The management scheme developed here is a modified form of the deterministic crowding algorithm [12]. Deterministic crowding was selected because of its simple, yet powerful, method of selection. The method requires no control parameters, and in addition, since the parents are randomly selected from the population, there is neither a need for fitness-based selection probability calculations (such as in fitness sharing [7]) nor large numbers of niche distance calculations (such as in crowding [5] and clearing [14]). The simplicity of the method also makes it suitable for possible hardware implementation.

The absence of control parameters, such as the niche radius used in fitness sharing and clearing, is of particular importance in this application. In [15], it is shown that a suitable choice of niche radius relies upon a knowledge of the structure of the fitness landscape. The complex one-to-many mapping of the CLUT makes this difficult to obtain.

The niche measure required by deterministic crowding to determine the similarity of individuals is commonly based upon properties of the phenotype of the individuals. This raises two difficulties in this application. Firstly, the length of the solution is not contained within the phenotype. Secondly, the length of the phenotype grows exponentially with the dimensions of the filter neighbourhood, and so, the approach will not scale well for larger neighbourhoods. An alternative approach would be to base the measure upon the genotype of individuals (i.e., the Hamming distance). However, the use of a too simple genotype measure will degrade the performance of the genetic algorithm [10]. The method presented

```

(1) Initialise and evaluate the fitness of the population
(2) repeat
(3) Randomly select two parents from the population
(4) if stagnation has occurred then
(5) Create a single offspring through concatenation with probability
     $\mu$ , or perform uniform parameterised crossover
(6) Apply duplication-rotation with probability  $\nu$ 
(7) else
(8) Create a single offspring using uniform parameterised crossover
(9) end if
(10) Apply point mutation
(11) Evaluate the fitness of the offspring according to (6.13)
(12) Determine the most similar parent
(13) if the offspring is fitter than the most similar parent then
(14) Replace the parent with the offspring
(15) end if
(16) until termination criteria are satisfied

```

ALGORITHM 6.1. An outline of the steady-state variable-length algorithm.

in Section 6.5, uses a weighted-genotype measure which is designed to reflect the sensitivity characteristics of the CLUT.

6.5. The genetic algorithm

The general structure of the genetic algorithm (shown in Algorithm 6.1) can be considered to be that of a variable-length form of the standard deterministic crowding method presented in [10, 11]. Standard deterministic crowding creates parental pairs by employing random selection without replacement, where selection is based upon the entire population. That is, all individuals in the population are used to form parental pairs and each individual is selected only once. Crossover is then applied to each pair to create two offspring. After mutation of the offspring, the niche measure is used to determine the distances between the offspring and their parents. The offspring and their parents are then paired in the configuration which results in the smallest distance sum. Finally, if the offspring is fitter than its parent, then the offspring replaces the parent in the population.

The modified steady-state variable-length algorithm differs in the following aspects.

- (1) Only one parental pair is selected from the population during each iteration, owing to the use of the steady-state scheme.
- (2) Only a single offspring is produced. This improves the matching of the distance between parent and offspring, since only a single distance is used instead of the sum of two distances.

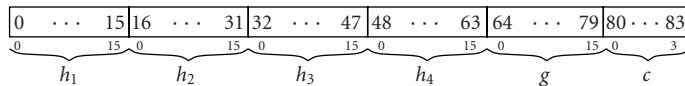


FIGURE 6.4. The structure of a single CLUT operator genotype.

- (3) When the progress of evolution begins to stagnate, the length of the offspring is increased through the use of constructive compositional operators (i.e., operators which can create offspring with lengths other than those of the parents).
- (4) The surviving individuals are immediately reinserted into the population prior to the next selection. This follows from the first point and allows offspring to take part in the selection process immediately.

The following sections give a detailed description of the steps given in Algorithm 6.1.

6.5.1. Genome structure

A single transform-connective pair (i.e., $C_n(\cdot) - \circ_n$, given in (6.3)) is uniquely determined within the genome of an individual, by 84 bits (as shown in Figure 6.4). The first 80 bits define the contents of the CLUT operator and the corresponding transform, and the remaining four bits specify the connective to be implemented by the postprocessor.

6.5.2. Initialisation

The population is initialised with individuals representing single CLUT operators. The bit strings representing each look-up table of an individual are randomly generated according to a probability density selected from a distribution of densities, which are uniformly distributed over the range $[0, 1]$. The motivation for choosing different densities for the five look-up tables is to allow specific individuals to focus on different areas of the structuring element. The associated connectives are selected from the list, given in Table 6.1, in a similar fashion.

6.5.3. The evolutionary operators and their progression-based application

As indicated, the application of the compositional operators adapts to the progress of evolution (steps 4–9) and only occurs after a period of stagnation. Stagnation is said to have occurred when the fitness of the population, measured as the fitness of the fittest individual within the population, no longer improves over an interval of 80 generations. When a stagnation phase is reached, the compositional operators are applied for 20 generations. If there continues to be no improvement in the fitness of the population, the cycle is repeated. The length of the two intervals were determined experimentally.

The probability that an individual evolutionary operator is applied varies with an estimate of the current average length of the individuals within the population (\bar{N}). The adaptation mechanism ensures that the number of applications made to an individual is approximately constant, and that it is independent of an individual's length. The evolutionary operators, mutation and crossover, are only applied to the CLUT section of the genome. A description of the individual operators and their applications is given below.

(1) *Mutation.* The single accretive operator within the algorithm is standard point-mutation and has an average application rate of two bit transitions per offspring.

(2) *Crossover.* The form of crossover (applied at steps 5 and 8 of the algorithm) is uniform parameterised crossover. The average rate is two crossover sites per offspring. The length of the resulting offspring is that of one of the parents and is determined by the final crossover site (see Figure 6.5).

(3) *Concatenation.* The concatenation operator concatenates the parents to form a single offspring. It replaces the crossover operation during stagnation (step 4) and an experimentally determined rate of application of $\mu = 0.00375$ has been used.

(4) *Duplication-rotation.* The duplication-rotation operator is applied to a randomly selected CLUT operator within the sequence of an offspring. The selected CLUT operator is copied, rotated, and inserted back into the sequence immediately following the position of the initial operator (see Figure 6.6). The form of the rotation is based upon the phenotype of the CLUT operator and aids in the creation of subsequences which implement rotationally invariant image transformations. The rotation is performed by circularly shifting the bit strings representing the contents of $h_1 \dots h_4$ and transforming the contents the bit string representing the look-up table g . The resulting effect is that the neighbourhood pattern implemented by the new CLUT operator corresponds to one of the four central 90 degrees rotations of the selected CLUT operator. The application rate of the operator, ν , is governed by $\nu = 0.015/\bar{N}$, where \bar{N} is an estimated average of the number of CLUT operators in an offspring.

6.5.4. The weighted-genotype niche measure

The niche measure, given in (6.6), employs two weights (α, β) to accentuate the relative effects of different sections of the genome and a third single weight (γ) to reflect the difference in length,

$$d(G_a, G_b) = \gamma \cdot |L_a - L_b| + \sum_{k=1}^{\min(L_a, L_b)} w_k \cdot (G_a(k) \vee G_b(k)). \quad (6.6)$$

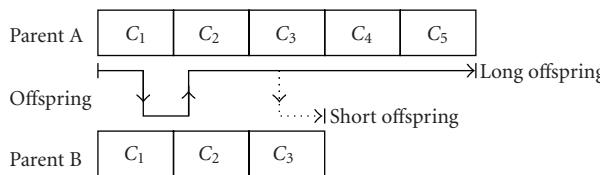


FIGURE 6.5. An illustration of variable-length crossover. The offspring will, in this case, take the length of the longer parent A. If there were to be a third crossover site, as indicated by the dashed trace, the resulting offspring takes the shorter length of parent B.

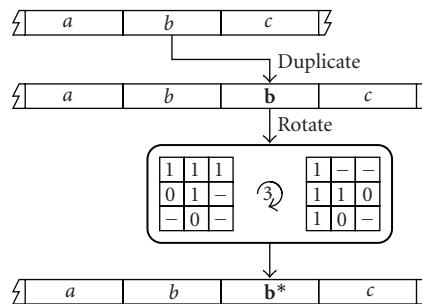


FIGURE 6.6. An example of the duplication-rotation operator, where the selected CLUT operator is rotated by 270° .

In (6.6), w_k is the position-dependent weighting factor for the k th bit of the genome given by (6.7), in which α is the weight corresponding to the bits of $g(\cdot)$ and β the weight for the connective,

$$w_k = \begin{cases} 1 & \text{if } k \in \{h_{n1} \dots h_{n4}\}, \\ \alpha & \text{if } k \in g_n, \\ \beta & \text{if } k \in c_n. \end{cases} \quad (6.7)$$

G_a and G_b are the two genomes between which the measure is to be made and L_a and L_b are their respective lengths measured in bits. The constant γ reflects the significance of the discrepancy in length between the two individuals and is chosen to be sufficiently large to assure the generation of length-based niches.

6.5.5. Fitness evaluation

The fitness measure consists of a product of three factors, each addressing one aspect of the solution. The first factor measures the correctness, the second applies a pressure favouring shorter solutions, and the third factor promotes individuals which implement elements of the solution not commonly found within the population. The correctness factor is also used as the progress measure when determining the presence of stagnation (step 4 in Algorithm 6.1).

An individual's correctness is measured as the number of target bits correctly implemented. The target function is represented as a vector of 512 Boolean values representing the desired output for all possible values in the neighbourhood. The mapping from neighbourhood to index within the vector is given by

$$i = \sum_{j=1}^9 2^{9-j} \cdot x_j, \quad (6.8)$$

where $x_1 \dots 9$ are the neighbourhood pixels shown in Figure 6.2. The correctness factor f_c is given in

$$f_c = \frac{1}{512} \sum_{i=0}^{511} s_i \vee t_i, \quad (6.9)$$

where s_i is the output of the CLUT sequence for input pattern i , and t_i is the corresponding target function value. A correctness factor of $f_c = 0$ indicates an entirely correct solution.

The second factor f_{pp} , given in (6.10), acts as a parsimonious penalty and is proportional to the logarithm of the length of the genome,

$$f_{pp} = 1 + c \cdot \ln(1 + N), \quad (6.10)$$

where N is the number of operators in the sequence and c is a positive scaling factor (taken as $c = 0.05$ in the experiments in Section 6.6).

The selection pressure, applied by f_c , will promote individuals which implement a larger cover of the target function. However, irrespective of the size of cover, an individual may still implement necessary elements of the target function. The use of f_c in isolation could lead to the removal of such individuals from the population.

In order to combat this effect, an adaptive memory-based niche focusing scheme, which continuously modifies the fitness landscape with the aim of promoting less common traits in the population, is also included within the fitness calculation. The scheme maintains a payoff vector of "recently seen" bit-pairs, calculated according to (6.11), and recomputed after each fitness evaluation,

$$p_i = \begin{cases} p_i \cdot \delta + (1 - \delta) & \text{if } s_i = t_i, \\ \max(p_i \cdot \delta, \check{p}) & \text{otherwise.} \end{cases} \quad (6.11)$$

In (6.11), p_i are the elements of the payoff vector, δ is the payoff time constant (which was set to $\delta = 0.99$ in Section 6.6), and \check{p} is the maximum payoff (which was set to $\check{p} = 0.1$ in Section 6.6).

The third fitness factor, f_s , corresponds to the payoff for the least commonly implemented target filter minterm, as shown in (6.12),

$$f_s = \min(p_i) : i \in [0 \dots 511]. \quad (6.12)$$

The total fitness, F , of an individual is calculated according to (6.13). If an individual correctly implements the target function, it receives a fitness determined solely by the number of operators, N , of the sequence. Otherwise, the fitness is given as the product of the three fitness factors,

$$F = \begin{cases} f_c \cdot f_{pp} \cdot f_s & \text{if } f_c \neq 0, \\ N \cdot \varepsilon & \text{otherwise.} \end{cases} \quad (6.13)$$

The constant ε is chosen such that $N \cdot \varepsilon < 1/512$ and is taken to be $\varepsilon = 10^{-5}$ in Section 6.6, (i.e., a fully correct individual is always better than an incorrect individual, for any practical value of N). The presence of ε ensures that competition continues between individuals after solutions to the target problem have been discovered.

6.6. Results

This section presents the results of a series of experiments, in which the algorithm has been applied to the optimisation of the median filter and three common morphological transformations. The results provide an analysis of an evolved solution and illustrations of the effects of diversity weights and population initialisation upon convergence and the degree of optimisation. The section begins by introducing the transforms used as a basis for the experiments.

6.6.1. The mapping of structuring element sequences

The three morphological image transforms, shrink, thin, and skeletonisation are commonly used in machine vision applications. Typically, the algorithms differ only in their choice of structuring element sequence and are implemented by iterating the sequences until there is no further change in the image.

The algorithms for shrink, thin, and skeletonisation given in [13] are based upon a staged application of two 3×3 structuring element sequences. During the first stage of the algorithm (known as the conditional stage), the structuring elements are used to mark pixels for conditional erasure. The second stage (known as the unconditional stage) selectively removes marked pixels from the image using an additional set of structuring elements. Table 6.2 shows the number of structuring elements required for each algorithm and the length of the shortest operator sequence evolved by the genetic algorithm, when the population was initialised with single CLUT individuals. Inspection of the evolved CLUT sequences shows that the algorithm was able to map 5–15 structuring elements to a single CLUT operator.

Figures 6.7(a) and 6.7(b) respectively illustrate the variation in the correctness factor and the staged growth of individuals during the evolution of a solution to the unconditional skeletonisation problem.

TABLE 6.2. The optimisation results for skeletonisation, thin, and shrink for $\alpha = 8$, $\beta = 1$, and an initial population of 400 single CLUT operator individuals.

Transform	Stage	Structuring elements	Sequence length
Skeletonisation	Conditional	40	4
	Unconditional	28	6
Thin	Conditional	46	5
	Unconditional	45	7
Shrink	Conditional	58	4
	Unconditional	37	7

6.6.2. The analysis of a particular solution

Figure 6.8 illustrates how a particular individual, which solves the unconditional skeletonisation set, implements the minterms of the target function. In Figure 6.8, each consecutive row corresponds to the minterms which result after the application of the respective operator in the sequence $C_{1\dots 6}$. The hashed squares represent the minterms which are currently implemented incorrectly, and the empty squares represent a correct implementation. The figures to the right of the illustration indicate the percentage of the total number of minterms, in the target function, which were correctly implemented at each point in the sequence.

The first operator implements 90.0% of the minterms correctly. Each operator thereafter in the sequence increases the number of correct minterms until the fifth operator is reached. The fifth operator decreases the total fraction of correct minterms but in doing so it also corrects minterms within other parts of the solution, which up until that point, had been incorrect. The change in the greedy nature of the algorithm, indicated by the fifth operator, is found in the majority of the solutions evolved by the algorithm. Figure 6.9 illustrates the processing stages for the evolved unconditional skeletonisation sequence applied to an image.

6.6.3. The effect of initial population length upon diversity

The effect of the initialised length of individuals upon the diversity of the evolving population is illustrated in Figures 6.10(a) and 6.10(b). Figure 6.10(a) shows the average individual diversities of the first six CLUT operators of the sequence, when evolving solutions to the unconditional skeletonisation problem with an initial population randomly seeded with individuals up to a maximum length of eight. Figure 6.10(b) shows the diversities of the same six CLUT operators, but for the case, where the population was initialised with individuals of length one. The figures show that a greater diversity in all operator positions was achieved when the population had been initialised with single CLUT individuals, and that this effect also continues after the settling period.

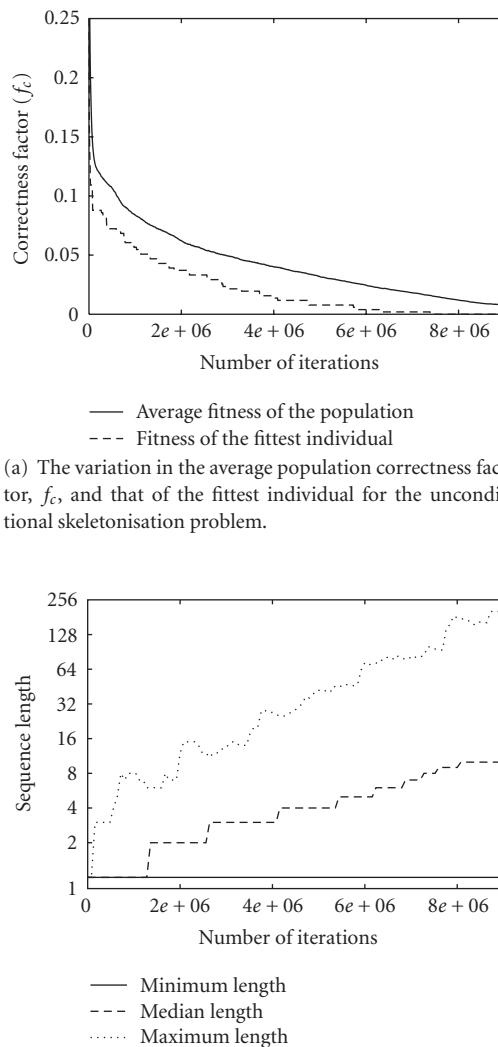


FIGURE 6.7. The progression of the correctness factor and growth in genome lengths of the population.

6.6.4. Variation in diversity parameters

Table 6.3 contains the results of a series of experiments used to investigate the variation in the weighted-genotype diversity parameters, α and β , upon the performance of the algorithm. The performance was measured in terms of the number of quasi-shortest solutions (i.e., the shortest solution as yet discovered by the algorithm). It can be seen from Table 6.3 that smaller β/α ratios result in a larger

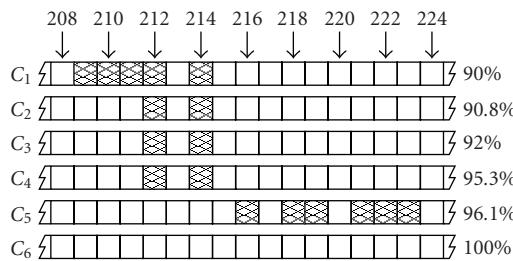


FIGURE 6.8. An illustration of a section of an evolved CLUT sequence showing the alteration in solution strategy.

number of short solutions. The experiments evolved solutions for the median filter and allowed evolution to continue for 2 million iterations beyond the point where the first solution was discovered (approximately one third of the total number of iterations).

The performance of the algorithm was then investigated with alternative forms of niche measure. The measures included the genotype Hamming distance, the phenotype Hamming distance, and a phenotype measure sensitive to the length of the CLUT operator sequence, calculated as in

$$d_{L\text{-phenotype}}(P_a, P_b) = 512 \cdot |N_a - N_b| + \sum_{i=0}^{511} P_a(i) \vee P_b(i). \quad (6.14)$$

The lengths of the respective solutions are shown in Table 6.4, together with the times to first solution measured in terms of the number of iterations of the algorithm. The results of the L -phenotype and weighted-genotype measures show a similar number of short solutions and a similar spread in the final length of the individuals. The phenotype measure has the shortest time to solution, but generates the longest solutions.

6.7. Discussion

In [10], genotype and phenotype deterministic crowding niche measures are analysed in terms of a number of smooth single variable function optimisation problems. The functions used within the study exhibit neither epistatic interaction nor neutral evolution. The results of the study motivated the use the phenotype measures, as they maintained diversity within the decoded parameter space.

Figure 6.11 shows the correlation between the phenotype and weighted-genotype measure taken from a population of individuals during the evolution of the median filter. From inspection, the values of the weighted-genotype parameters which best correlate with the phenotype involve intermediate connective weights (i.e., β). This is in agreement with the results of weight variation (shown in Table 6.3) in evolutionary runs, where it can be seen that the best performance (measured in terms of sequence length) was achieved when the connective weight

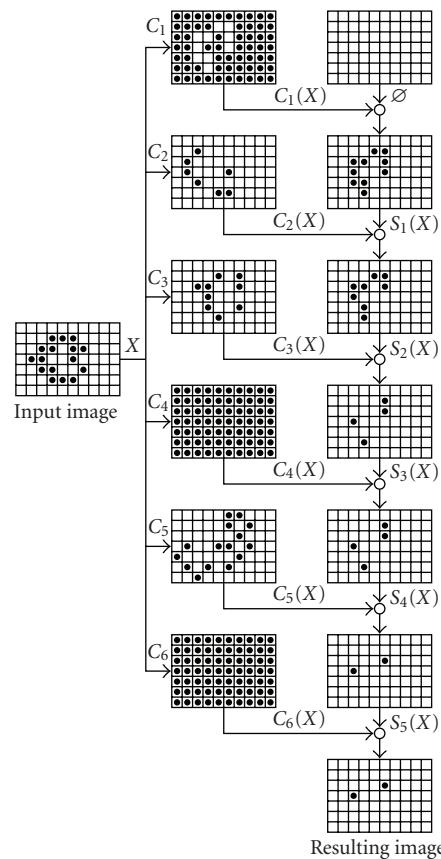


FIGURE 6.9. The figure illustrates the processing of an image, showing the six steps of the evolved unconditional skeletonisation operator. The input image is taken from the output of the preceding conditional skeletonisation processing stage. The input image contains the pixels which potentially are to be erased. The resulting image contains two pixels, which the second stage of the algorithm has selected for protection from erasure.

was relatively low. The reasons can be found in two elements of the problem structure.

Firstly, the connective element of the genome represents only ten possibilities, and once created, these remain unchanged by the evolution. A greater connective weight tends to promote genetic diversity based only upon these few values and to the detriment of the CLUT operator. The limited variation in the connective alleles results in low phenotype diversity.

Secondly, the sources of neutral evolution and epistatic interaction are localised within the gene representing the look-up table g . This gene controls how the genes of the first layer look-up tables are expressed in the phenotype. Thus, an increase in the relative importance of α promotes diversity in both g and h ,

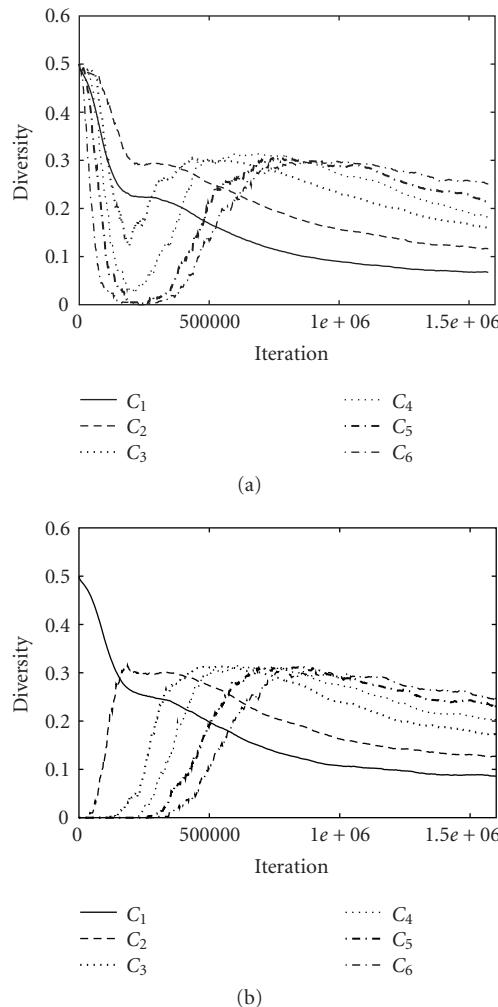


FIGURE 6.10. The two graphs show the separated genotype diversity for the first six CLUT operators during evolution. In (a), the initial individual lengths are uniformly distributed between one and eight CLUT operators and in (b), the population is initialised with single CLUT individuals.

reduces niching based upon the connective, and provides greater diversity in the phenotype and the resulting decoded parameter space.

The relative weights of g and h are derived from the product of their relative sensitivities (i.e., two to one as shown in Figure 6.3) and their relative lengths within the genome (i.e., four to one). The analysis in [10] compared the performance of a phenotype measure with that of a uniformly weighted genotype Hamming distance and demonstrated the improved time to solution of the phenotype measure. A similar effect is shown in Table 6.4 for this application, however, the lengths of the solutions are the longest. The further improvement achieved by the

TABLE 6.3. An illustration of the effect of the variation of the parameters α and β upon the number of short solutions evolved by the algorithm. Each result is calculated from 199 evolutions of the median filter problem. Once a solution was evolved, the algorithm was allowed to continue for further 2 million iterations. A dash indicates that no short solutions were evolved.

$\beta \setminus \alpha$	1	2	4	8	16
1	2	5	3	3	5
4	5	3	2	5	2
16	1	1	1	1	5
64	1	—	2	4	1
256	1	—	2	1	2
1024	1	—	2	1	2
4096	1	—	2	1	2

TABLE 6.4. An illustration of the range of solution lengths evolved for each of the four measures, together with their associated “times to first solution,” measured in terms of the number of algorithm iterations. Each of the values is calculated from a series of 199 runs.

	Length			Time to solution (10^6)		
	Min	Median	Max	Min	Median	Max
Genotype	4	6	12	2.28	3.21	7.14
Phenotype	5	8	20	0.85	1.85	> 20.0
<i>L</i> -phenotype	4	6	10	2.57	4.96	> 20.0
WG (8,1)	4	6	8	2.56	3.71	15.75

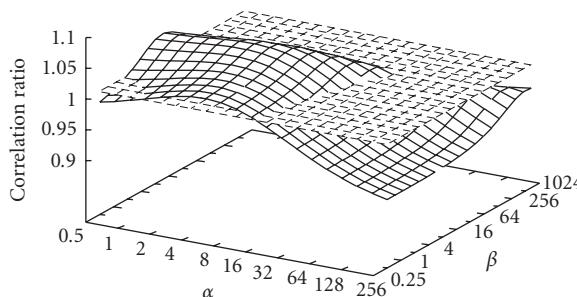


FIGURE 6.11. Illustrating the correlation between the weights of the weighted-genotype measure and the phenotype of individuals gathered from a population evolving a solution to the median filter problem.

L-phenotype measure over the phenotype measure is attributable to the increased number of niches created when a measure also accounts for the length of the individual. The hierarchical niching created by the differing weights in the weighted-genotype measure results in a similar number of short solutions as the *L*-measure but provides a shorter mean time to solution, without the additional cost to evaluate the phenotype.

The adaptive niche focusing technique developed for the algorithm provides a nonstationary fitness landscape which effectively guides the search toward, what are currently, the least common traits of the solution found in the population. As the algorithm evolves new elements to the solution, the payoff vector is modified and the fitness of an individual changes with the progression of evolution. The complex interplay between the payoff time constant, maximum payoff, population size, and the convergence of the algorithm required that the payoff time constant be determined by experimentation.

The weights of the weighted-genotype measure are assigned to genes and not to the alleles within the genome. Furthermore, the weights are constant and so are never modified to reflect the change in the fitness landscape. Based upon the evidence provided in Table 6.3, it is unlikely that globally optimal weights exist, and that locally optimal weights depend upon not only the target filter, but also the initial population and current stage of evolution. Further work will investigate alternative adaptive weighted-genotype measures which extend the assignment of weights to the level of the allele.

Bibliography

- [1] C.-W. Ahn and R. S. Ramakrishna, "A genetic algorithm for shortest path routing problem and the sizing of populations," *IEEE Transactions on Evolutionary Computation*, vol. 6, no. 6, pp. 566–579, 2002.
- [2] M. Bertozzi, A. Broggi, and A. Fascioli, "Vision-based intelligent vehicles: state of the art and perspectives," *Robotics and Autonomous Systems*, vol. 32, no. 1, pp. 1–16, 2000.
- [3] A. Broggi, G. Conte, F. Gregoretti, C. Sansòè, R. Passerone, and L. M. Reyneri, "Design and implementation of the PAPRICA parallel architecture," *The Journal of VLSI Signal Processing*, vol. 19, no. 1, pp. 5–18, 1998.
- [4] E. K. Burke, S. Gustafson, and G. Kendall, "Diversity in genetic programming: an analysis of measures and correlation with fitness," *IEEE Transactions on Evolutionary Computation*, vol. 8, no. 1, pp. 47–62, 2004.
- [5] K. A. de Jong, *An analysis of the behavior of a class of genetic adaptive systems*, Ph.D. thesis, University of Michigan, Ann Arbor, Mich, USA, 1975.
- [6] A. L. Fisher, "Scan line array processors for image computation," in *Proceedings of the 13th Annual Symposium on Computer Architecture*, pp. 338–345, Tokyo, Japan, 1986.
- [7] D. E. Goldberg and J. Richardson, "Genetic algorithms with sharing for multimodal function optimization," in *Proceedings of the 2nd International Conference on Genetic Algorithms (ICGA '87)*, pp. 41–49, Lawrence Erlbaum Associates, Cambridge, Mass, USA, July 1987.
- [8] J. H. Holland, *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence*, University of Michigan Press, Ann Arbor, Mich, USA, 1975.
- [9] P. P. Jonker, E. R. Komen, and M. A. Kraaijveld, "A scalable, real-time, image processing pipeline," *Machine Vision and Applications*, vol. 8, no. 2, pp. 110–121, 1995.
- [10] S. W. Mahfoud, "Crowding and preselection revisited," in *Proceedings of the 2nd Conference on Parallel Problem Solving from Nature*, pp. 27–36, Brussels, Belgium, September 1992.
- [11] S. W. Mahfoud, "Crossover interactions among niches," in *Proceedings of the 1st IEEE International Conference on Evolutionary Computation (ICEC '94)*, vol. 1, pp. 188–193, Orlando, Fla, USA, June 1994.
- [12] S. W. Mahfoud, "A comparison of parallel and sequential niching methods," in *Proceedings of the 6th International Conference on Genetic Algorithms (ICGA '95)*, L. J. Eshelman, Ed., pp. 136–143, Morgan Kaufmann, Pittsburgh, Pa, USA, July 1995.

- [13] W. K. Pratt, *Digital Image Processing*, John Wiley & Sons, New York, NY, USA, 2nd edition, 1991.
- [14] A. Pétrowski, "A clearing procedure as a niching method for genetic algorithms," in *Proceedings of IEEE International Conference on Evolutionary Computation (ICEC '96)*, pp. 798–803, Nagoya, Japan, May 1996.
- [15] B. Sareni and L. Krahenbuhl, "Fitness sharing and niching methods revisited," *IEEE Transactions on Evolutionary Computation*, vol. 2, no. 3, pp. 97–106, 1998.
- [16] I. P. W. Sillitoe, A. K. Magnusson, and M. G. Nordahl, "Clutter: a FPGA based binary image processor for embedded systems," in *Proceedings of the 8th International Conference on Mechatronics*, pp. 828–837, Enschede, The Netherlands, June 2002.
- [17] I. P.W. Sillitoe and M. Tombak, "A compact look-up table structure for low-level binary image processing," *Real-Time Imaging*, vol. 4, no. 3, pp. 203–210, 1998.
- [18] P. Soille, "On morphological operators based on rank filters," *Pattern Recognition*, vol. 35, no. 2, pp. 527–535, 2002.
- [19] R. Stojanovic, P. Mitropoulos, C. Koulamas, Y. Karayannidis, S. Koubias, and G. Papadopoulos, "Real-time vision-based system for textile fabric inspection," *Real-Time Imaging*, vol. 7, no. 6, pp. 507–518, 2001.

Ian P. W. Sillitoe: Scottish Association for Marine Science, Dunstaffnage Marine Laboratory, Dunbeg, PA37 1QA Oban, Scotland

Email: ian.sillitoe@sams.ac.uk

Andreas K. Magnusson: School of Engineering, University College of Borås, SE-501 90 Borås, Sweden

Email: andreas.magnusson@hb.se

7

Autonomous model-based corner detection using affine evolutionary algorithms

Gustavo Olague and Benjamín Hernández

7.1. Introduction

Photogrammetry is the science, and art, of determining the size and shape of objects as a consequence of analyzing images recorded on film or electronic media. Computer vision can be understood as the science of obtaining reliable, accurate, and useful information from images in order to execute and complete tasks devoted to perceiving, sensing, and interacting with the world around a machine vision system. It is clear from those definitions that corner detection is a basic operation in photogrammetry and computer vision. A great deal of effort has been spent by the computer vision and photogrammetric communities on this problem [1–3, 6, 8, 11, 13, 15, 16, 18, 19, 21–23, 29–32] and in particular on the problem of edge detection [4, 5, 17]. The problem of detecting the exact point that describes the corner position in the case of a bandlimited system should be approached carefully. This problem is of main concern for high-accurate reconstruction. High-accurate corner extraction is a complex process due to several factors: (1) the attitude, position, and orientation of the camera with respect to the object; (2) the interior orientation of the camera; (3) the fluctuations of the illumination; and (4) the camera optics [26].

Several approaches to the problem of detecting these feature points have been reported in the literature over the years. Some approaches work within pixel resolution as the Kitchen and Rosenfeld corner detector [16], the Harris detector [15], and the interest operator of Moravec [21]. Within the photogrammetric literature, the seminal approach described by Gruen [13] introduces least squares as a general technique for all kind of data-matching problems. This approach works directly on the gray-level image. Rohr [28] studied the displacement of the corner location using an analytical corner model, which is convolved with a Gaussian function in order to model the blur. Deriche and Giraudon [6] have developed a similar analytical study of corner models using the linear Gaussian scale space. They show that the local maximum in the image moves in the scale space along the bisector line that passes through their definition of the exact corner position.

A print edition of this book can be purchased at

<http://www.hindawi.com/spc.8.html>

<http://www.amazon.com/dp/9774540018>

However, the displacement of the local maximum with respect to the initial location of the corner depends on the angle of the corner [23]. This problem turns their criterion not suitable as it is not invariant to camera placement.

This chapter is devoted to the modeling of corner features using affine evolutionary algorithms. A novel parametric corner modeling based on a unit step edge function (USEF) is developed in order to define straight-line edges. The optical and physical characteristics of the image acquisition system are modeled by a distribution function that is simple, yet robust enough to encapsulate in one single equation the whole image projection of complex corners. The process of fitting the model to the image intensities is enhanced by searching model parameters with a global optimization technique using the least-squares criterium. The objective of this paper is to show a robust and reliable L-corner detector based on least-squares modeling of a *unit step edge function* fitted to window-image data using a global optimization technique. Our approach is motivated by the idea of avoiding the two-step process of convolving the proposed model with a Gaussian that is normally applied in previous research. Instead, we work with a model that takes into account directly the level of blurring found in bandlimited systems like the CCD cameras [24]. Moreover, the quality of detection relies on the approximation to the initial position estimation. Parametric approaches are suitable for highly accurate localization. This work reports the first study about which kind of approach should be applied to improve the accuracy while eliminating the problem of initial parameter estimation. This paper reports the performance of three different algorithms to measure L-corner positions in real images. Experimental results show the superiority of our L-corner model in real-image detection. A comparison with downhill simplex and simulated annealing using several levels of noise was performed to show the advantages and disadvantages of the evolutionary algorithm.

This chapter is organized as follows. Section 7.2 presents the main concepts used to classify all kind of complex corners. Section 7.3 introduces our parametric-based model, which is used to build complex corners, including the L-corner and vertex models. Section 7.4 introduces the idea of studying modeling of data as an optimization process. This approach provides best fit parameters, as well as the accuracy of the parameters. A novel evolutionary representation is introduced using the concept of affine transformation. An evolutionary algorithm is applied to improve the initial parameter estimation based on an affine transformation in order to enhance the evolutionary algorithm representation. The affine representation encapsulates within a single algebraic form the two main operations, mutation, and crossover of an evolutionary algorithm. Finally, experimental results are provided to show the efficiency of the affine evolutionary algorithm compared to downhill simplex and simulated annealing.

7.2. Corner classification

A corner model should be able to describe a number of attributes such as position or location, angle of aperture, orientation, edge shape, edge texture, contrast,

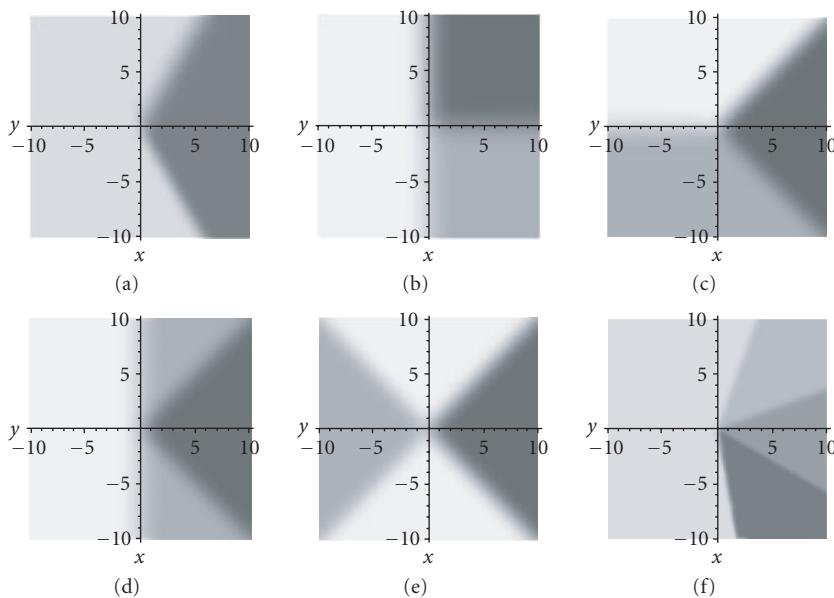


FIGURE 7.1. Corner classification according to the kind of border union. (a) L-corner, (b) T-corner, (c) Y-corner, (d) K-corner, (e) X-corner, (f) vertex. All figures were created with the USEF model.

edge profile, sharpness, color junction type, and size. In particular, those attributes could be grouped within the following three general properties.

- (1) Morphological characteristics. This set of properties are related to the qualitative aspects (texture, color, shape) describing the general exterior characteristics.
- (2) Geometrical characteristics. This group is related to the general shape produced by the edges composing the corner. These geometrical properties are able to describe the location of a corner with respect to a given coordinate system.
- (3) Physical characteristics. This set of attributes are related to the physical properties produced by the digital system (sensor, camera lens) when sampling a 3D scene.

7.2.1. Corner morphology

Morphological properties are classified according to the shape and number of edges defining a corner.

- (i) *L-corner*. It is generated when two straight-line edges join into a single point creating two gray zones, see Figure 7.1(a). The L-corner is the simplest corner structure.
- (ii) *T-corner*. It is produced when one edge joins other two edges out of its point of convergence. Normally, the edges create straight angles. In a T-corner three gray zones exist.

- (iii) *Y-corner*. Also known as “arrow head,” it is produced when three edges join into a common point, see Figure 7.1(c). In this kind of corner there are three different gray zones.
- (iv) *K-corner*. This is composed by the union of an L-corner with a region that does not belong to the end of a third edge, see Figure 7.1(d). As a result, the T-corner is a special case of a K-corner. The angle of the L-corner is about 90° , which is joined by a third edge at some nonextreme region.
- (v) *X-corner*. This is produced when four edges are joined within a region that does not belong to the end of those edges, see Figure 7.1(e). This kind of structures can also be seen as four edges that meet in the same point and the angles of the nonadjacent edges are equal.
- (vi) *Vertex*. This is produced when more than two edges are joined at a common point. The number of gray zones in a vertex is equal to the number of edges, see Figure 7.1(f). The T-corner, Y-corner, K-corner, X-corner are different kinds of vertex. The term vertex is strictly employed in this work. Hence, there are only two kinds of corners: the L-corner and the vertex.

7.2.2. Corner geometry

Geometrical corner properties are described by the set of straight-line edges and the curvature that is produced in the union of all edges. In general, all image acquisition systems provide images with a certain degree of blurring, because such devices are bandlimited systems. Therefore, it is suitable to estimate the level of uncertainty in order to obtain the set of parameters defining the position, orientation, and precise corner location. The uncertainty is related to the curvature within the union edge point. Corner modeling is based on the least squares fitting of the proposed L-corner or vertex model to the image intensities. As a result, we obtain the best fit parameters, as well as the covariance of each adjusted parameter. There are three geometrical properties.

- (i) *Angle of aperture*. A corner could be seen as a structure centered on a coordinate system. The angle between edges with respect to a coordinate system could be used to characterize the corner model. In the case of an L-corner, the aperture angle of the corner is composed of two edges.
- (ii) *Corner position*. This term refers to the point where the corner is located. There are many theoretical developments about its definition:
 - (1) techniques based on border points;
 - (2) techniques based on geometrical properties; and
 - (3) techniques based on parametric models.

Figure 7.2, shows three different criteria to localize the corner. P_1 denotes the intersection between r_1 and r_2 straight lines. If the corner is well defined describing a sharp profile, then the corner position could be considered as P_1 . On the other hand, P_3 defines the corner location that is obtained after computing the maximum planar curvature. Some approximations compute the border points, which

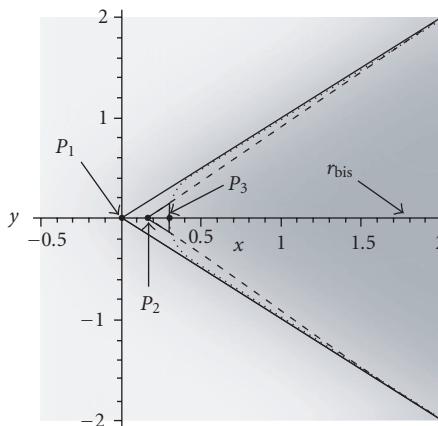


FIGURE 7.2. This graph shows several L-corner localization criteria that are reported in the literature.

are then used as a chain of points. Other approaches such as interpolation are used to compute the intersection of each asymptotic line in order to define P_2 , which is in general localized between P_1 and P_3 . In this work, a new corner criterion for localization has been introduced. This corner criterion considers the case of a rectangular CCD in order to estimate the nonuniform blur factor of a pulnix camera [23].

- (iii) *Corner orientation.* The coordinates of the corner point, as well as the direction of the straight lines define the orientation of the corner. The orientation could also be defined by the line that bisects the aperture angle and cross the corner point. Figure 7.2 shows the straight line r_{bis} which is the bisector line of the corner. In this case, the corner is symmetrical with respect to the y -axis.

7.2.3. Physical properties of a corner

The physical characteristics of an image-corner give a significant description about the quality of the corner. These characteristics describe numerically concepts related to the illumination of the three-dimensional scene, the quality of optical parameters such as focusing, and the distortion produced by the shape and size of the digital sensors. This physical characteristics are related to the concept of blurring. The term blurring refers to the level of fuzziness in the image. It indicates if the border is well defined. In other words, if the border profile can be seen clearly. Blurring is the main factor in the level of uncertainty of the corner location. In the case of an L-corner, the measurement of corner location is possible, as long as we can discern between the two gray levels. The factors producing the blurring are as follows.

- (1) *Focus.* If a section of the 3D scene is located out of the focal plane of the image acquisition system, then the image is out of focus. This phenomenon produces a continuous blurring on the image. Thus, the location of

the camera with respect to the object and the optical system plays a key role in the process of corner location.

- (2) *Aperture.* The aperture of the camera lens is finite and it contributes to the problem of bandlimited systems found on digital cameras. This problem produces also a homogeneous blurring.
- (3) *Variation on illumination.* The variation of illumination on the scene produces an irregular blurring around the whole image. However, if the interest region is relatively small, the blurring could be considered as homogeneous.
- (4) *Sampling.* A scene is sampled typically with a CCD sensor. The CCD is composed of a matrix or array of photo sensible elements, where each element represents a pixel in the digital image. It is common to find nonsquare elements. The rectangular pixels produce two different blur factors along the two main directions of the image sensor. Our analytical corner model proposed in this work characterizes completely this phenomenon. We have not found a previous work in the literature that states this important characteristic of digital cameras.

7.3. Unit step edge function model

Considering an image coordinate system $I(x, y)$ and an unknown set of parameters $P = (p_1, \dots, p_n)$, the unit step edge function is constructed based on the error function definition.

Definition 7.1 (error function). The error function, also called Gaussian probability integral, is a special case of the incomplete gamma function, and is obtained directly from C compilers. Its definition is

$$\text{erf}(x) = \frac{2}{\sqrt{\pi}} \int_0^x e^{-t^2} dt. \quad (7.1)$$

The function has the following limiting values and symmetries:

$$\text{erf}(0) = 0, \quad \text{erf}(\infty) = 1, \quad \text{erf}(-x) = -\text{erf}(x). \quad (7.2)$$

According to the above definition, we can derive a new function dividing the error function by 2 and adding half of a normal distribution in order to obtain a distribution function as follows:

$$F(x) = \frac{1}{\sqrt{\pi}} \int_0^x e^{-t^2} dt + \frac{1}{\sqrt{2\pi}} \int_{-\infty}^0 e^{(-1/2)t^2} dt = \frac{\text{erf}(x)}{2} + \frac{1}{2}. \quad (7.3)$$

By replacing x appropriately we can derive the USEF definition along the x -axis.

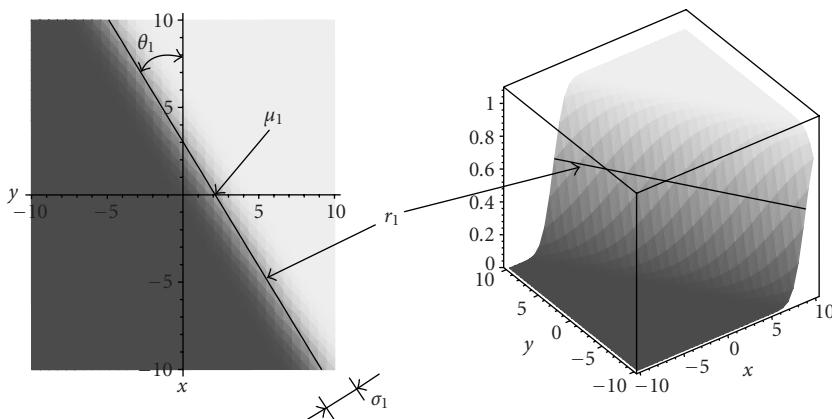


FIGURE 7.3. The straight line and its main parameters superimposed to the unit step edge function model $U_x(I, P_x)$.

Definition 7.2 (unit step edge function). Let the image coordinates and the set of unknown model parameters be denoted by $I = (x, y)$ and $P_x = (p_{x1}, \dots, p_{xn})$, respectively. The unit step edge function is represented as follows:

$$U_x(I, P_x) = \pm \frac{1}{\sigma_1 \sqrt{2\pi}} \int_0^x e^{-(t-y \cdot \tan(\theta_1) - \mu_1)^2 / 2\sigma_1^2} dt + \frac{1}{2}, \quad (7.4)$$

where the image coordinates are in the interval $[-m, m]$; the central point μ_1 designates the position x of the line that crosses along the y -axis; μ_1 lies in the interval $[-m, m]$; the rotation θ_1 is made clockwise about the (positive) y -axis; θ_1 designates the orientation of the edge model to be fitted to the image within the interval $-\pi/2 < \theta_1 < \pi/2$; finally, a scaling factor σ_1 that characterizes the amount of blur introduced by the discretization process is included. σ_1 lies in the interval $[0, m]$. The unit edge function describes a distribution function that increases steadily from 0 to 1 with respect to the x -axis.

The graphical model of the USEF is the three-dimensional step edge shown in Figure 7.3. This model describes completely the 2D intensity variations within a single equation instead of the two-step process of convolving an ideal-shaped gray-value structure with a Gaussian filter as is normally done. In this way, it is straightforward to scale the model to the 2D intensity variations using the operations of addition and multiplication as follows:

$$U'_x(I, P_x) = U_x(I, P_x)A + B, \quad (7.5)$$

where A represents the distance between the lower and upper gray levels and B represents the lower gray value, also called here floor level. The unit step edge function $U_y(I, P_y)$ with respect to the y -axis is represented in a similar way, where all intervals of the variables remain the same and μ_2 designates the position y of

the line that crosses the x -axis. The rotation θ_2 designates the orientation of the unit step edge model in the y -direction. $U_y(I, P_y)$ can be evaluated numerically using the Gaussian error function as follows:

$$U_y(I, P_y) = \pm \frac{1}{2} \operatorname{erf} \left(\frac{(y - x \cdot \tan(\theta_2) + \mu_2)}{\sigma_2 \sqrt{2}} \right) + \frac{1}{2}. \quad (7.6)$$

Hence, the USEF $U_y(I, P_y)$ is characterized by an r_2 straight line along its main direction. The straight-line equation is obtained from the numerator in the argument of the exponential function, see Figure 7.3.

7.3.1. L-corner modeling

L-corners are generated when two straight-line edges join into a single point creating two homogeneous gray zones with different intensities, see Figure 7.4. This work proposes a new corner modeling based on the USEF model. In order to obtain a corner unit function (CUF), two USEFs are multiplied as follows:

$$M'_L(x, y, \vec{P}) = U_x(I, P_x) \cdot U_y(I, P_y) \cdot A + B. \quad (7.7)$$

The structure generated by (7.7) is known in the literature as the “L-corner.” The parameters $\sigma_1, \theta_1, \mu_1, \sigma_2, \theta_2, \mu_2, A$ and B , represent the physical and geometrical contours of an L-corner. Therefore, in order to obtain the corner model, we simply multiply both USEFs. In summary, our model is based on an analytical expression with the following characteristics.

- (1) Each edge on the corner has different levels of blurring. This is physically produced by the nonsquare CCD pixels of the Pulnix 9701 camera. Hence, the CUF models the degree of blurring along each edge.
- (2) Each angle is independent. Therefore, there is not any restrictions with respect to the acute or obtuse angles within the corner.
- (3) The corner moves freely around the explored window. We obtain the position and orientation of the corner around any point within the studied window.
- (4) The gray levels are self-adjusted inside and outside the corner.

7.3.2. Extraction of multiple features

Considering the above procedure for building L-corners, it is possible to model arbitrarily complex gray-value structures in terms of the unit step edge function by means of simple addition and multiplication operations. The total number of parameters used by the model is in general

$$n = 3 + 2N + O, \quad (7.8)$$

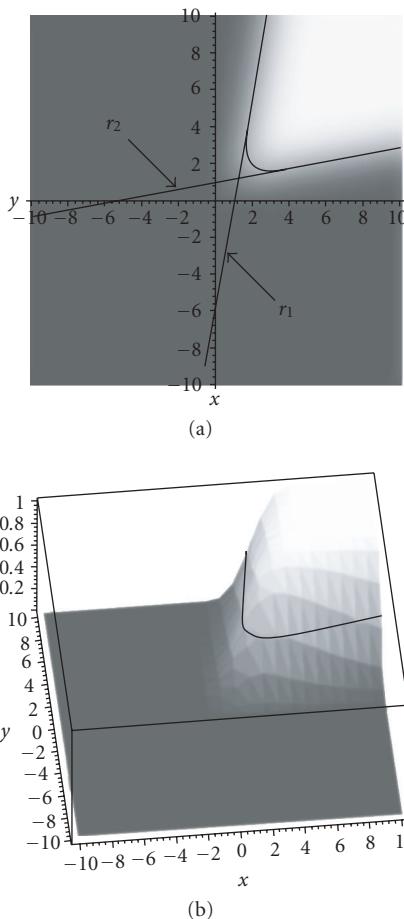


FIGURE 7.4. Corner unit function $M_L(x, y, \vec{P})$ built from two USEFs. (a) Top view of the CUF model showing both straight lines, r_1 and r_2 , along both edges. (b) Three-dimensional view of the corner, as well as the central contour curve.

where the first three, in the case of the L-corner, is given by the amount of blur σ and the lower and upper gray values A and B , respectively; N represents the number of step edge models, and O specifies the number of operations used to represent the feature. In fact, the first three should be increased if we take into account different blurs for each main direction and multiple gray values when considering more complex models. For example, the vertex model needs an extra gray level, and in total we need twelve parameters to model the corner, see Figure 7.5. A vertex model (VUF) can be easily obtained from an L-corner model (CUF) and a third USEF as follows:

$$V'(I, P) = U_x(I, P_x) \cdot U_y(I, P_y) \cdot A + U_z(I, P_z) \cdot B + C. \quad (7.9)$$

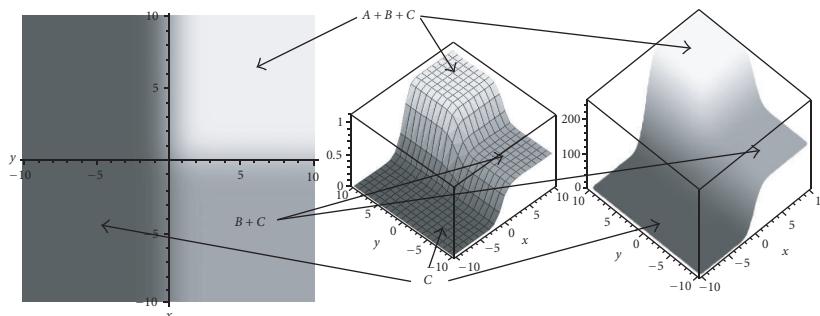


FIGURE 7.5. The vertex model can be easily obtained from an L-corner model and a third USEF.

7.4. Modeling of data and multidimensional optimization

Experimental science is devoted to fitting a model that depends on adjustable parameters to a given set of observations. The common approach is to select or design a merit function that measures the agreement between the data and the model with a particular choice of parameters. The model parameters are then adjusted to achieve a minimum in the merit function, yielding a set of best-fit parameters. The adjustment process is basically a problem of minimization in many dimensions. Finding the set of parameters that takes the function to a minimum or a maximum value is considered an optimization problem.

Definition 7.3 (global optimization). In general, an optimization problem requires finding a set of $\vec{P} \in S$, where S is a bounded set on \mathbb{R}^n , such that a certain quality criterion $f : S \rightarrow \mathbb{R}$, typically called the objective function, is minimized or equivalently maximized. Without loss of generality, it is sufficient to consider only minimization tasks, since maximizing $f()$ is equivalent to minimizing $-f()$. The problem then is to find a point $\vec{P}_{\min} \in S$ such that $f(\vec{P}_{\min})$ is a global minimum on S . More specifically, it is required to find an $\vec{P}_{\min} \in S$ such that

$$\forall \vec{P} \in S : f(\vec{P}_{\min}) \leq f(\vec{P}). \quad (7.10)$$

The tasks of maximization and minimization are trivially related to each other as one being the inverse of the other. An extremum (maximum or minimum) can be either global, truly the best solution, or local, the best around a neighborhood. Finding a global extremum is, in general, a very difficult problem. Moreover, in fitting data usually the merit function is not unimodal, with a single minimum, which makes the problem harder. On the other side, there are important issues that are beyond the mere finding of best fit parameters. Data are generally not exact! Data are subject to measurement errors. Thus, typical data never fit exactly the model that is being used, even when the model is correct. It is customary to assume that the measurements are independent random variables. Each measurement $(f(x_i, y_i), x_i, y_i)$ have a mean and a standard deviation. Fitting such a model

to the data is carried out through the well-known technique of least squares. The approach is to define an χ^2 merit function and determine the best fit parameters by its minimization. Because of nonlinearities, the minimization should proceed iteratively. Given an initial trial solution, sufficiently close to the minimum for the parameters; the process improves the trial solution iteratively until χ^2 stops, or effectively stop decreasing. Our approach is to apply a global optimization technique using the least squares method as a local process in order to improve the search of the global optimum. Moreover, as a by-product of the minimization, the covariances of the parameters are obtained.

7.4.1. Modeling L-corners as an optimization problem

The above analysis suggests that a global optimization technique can be used to solve the problem of guessing the initial parameters. Thus, corner's localization is obtained by fitting our parametric model to the image intensities. Estimates for the model parameters $\vec{P} = (p_1, \dots, p_n) \in \mathbb{R}^2$ are found by minimizing the squared differences between the (nonlinear) model function and the considered gray values:

$$Q = \chi^2 = F(\vec{P}) = \sum_{i=1}^m \sum_{j=1}^m [I(u_i, v_j) - M'_L(x_i, y_j, \vec{P})]^2. \quad (7.11)$$

The intensities and the function values of the model in the considered image area are $I(u_i, v_j)$ and $M'_L(x_i, y_j, \vec{P})$, respectively. Previous approaches used by Rohr [29] applied the method of Powell utilizing only function values or used the method of Levenberg-Marquardt (Press et al. [27]) incorporating partial derivatives of the model function in order to reduce the computation time. However, a drawback presented on these approaches is that the identification result relies on the initial parameter values and as usual with nonlinear cost functions, in general, we cannot guarantee to find the global minimum. This problem is studied in this work using an evolutionary algorithm due to the success achieved on this kind of problems. In summary, n is the number of parameters to minimize in our model. $m = 2w + 1$ defines the size of the input data. $\vec{P} = (\sigma_1, \mu_1, \vartheta_1, \sigma_2, \mu_2, \vartheta_2, A, B)$ are the parameters of M'_L that describe the behavior of our L-corner. $M'_L(x_i, y_j, \vec{P})$ is the corner model evaluated at the \vec{P} parameters on the model coordinate system. $I(u_i, v_j)$ are the intensity values of an image in a gray scale, which is a square subimage of size $m \times m$ pixels within the entire image. $F(\vec{P})$ is the χ^2 estimator. Equation (7.11) includes two coordinate systems: the image coordinate system (u, v) and the model coordinate system (x, y) , which are different.

7.4.2. Genetic algorithms for function optimization

Evolutionary algorithms are considered a rich paradigm for global optimization. Previous methodologies as the *downhill simplex method* and *simulated annealing*

are well-known techniques for multidimensional optimization [27]. This section is devoted to our affine evolutionary algorithm for global optimization. Currently, evolutionary algorithms for numerical optimization use real-code parameters for which a set of special transformations has been developed. In real coding implementation, each chromosome is encoded as a vector of real numbers of the same length. Several crossover operators have been introduced under the name of arithmetical operators. The arithmetical operators are built by borrowing the concept of linear combination of vectors from the area of convex sets theory. Generally, crossover produces an offspring, which is calculated from the weighted average of two vectors \vec{y}_1 and \vec{y}_2 as follows:

$$\begin{aligned}\vec{y}'_1 &= \lambda_1 \vec{y}_1 + \lambda_2 \vec{y}_2, \\ \vec{y}'_2 &= \lambda_2 \vec{y}_1 + \lambda_1 \vec{y}_2;\end{aligned}\quad (7.12)$$

if the multipliers are restricted to

$$\lambda_1 + \lambda_2 = 1, \quad \lambda_1 > 0, \quad \lambda_2 > 0, \quad (7.13)$$

the weighted form is known as convex combination. If the nonnegativity condition on the multipliers is dropped, the combination is known as affine combination. Finally, if the multipliers are simply required to be in real space, the combination is known as a linear combination [12]. Another operator is known under the name of dynamic mutation, also called nonuniform mutation, introduced by Janikow and Michalewicz [20]. Dynamic mutation is designed for fine-tuning capabilities aimed at achieving high precision. Given a parent \vec{y} , if the element y_k is selected for mutation, the resulting offspring is $\vec{y}' = [y_1, \dots, y'_k, \dots, y_n]$, where y'_k is randomly selected from the following two possibilities:

$$y'_k = y_k + \Delta(t, y_k^U - y_k) \quad (7.14)$$

or

$$y'_k = y_k - \Delta(t, y_k - y_k^L), \quad (7.15)$$

where

$$\Delta(t, y) = yr \left(1 - \frac{t}{T}\right)^b. \quad (7.16)$$

The function $\Delta(t, y)$ returns a value in the range $[0, y]$ such that the value approaches 0 as t increases. This property causes the operator to search the space uniformly initially, when t is small, and very locally at later stages. t is the generation number, b is a parameter determining the degree of nonuniformity, and r is a random number between $[0, 1]$. It is possible for the operator to generate an offspring which is not valid. In such a case, we can reduce the value of the random number r .

7.4.3. A novel evolutionary representation

The operations of crossover and mutation can be encapsulated into a single complex transformation as follows. In order to handle affine geometry algebraically, we have to characterize the line i by an invariant equation, and we will suppose that this equation is $y_0 = 0$. Since the points of i are now regarded as ideal points, no point with $y_0 = 0$ is actual, and this means that we can represent the actual points of the affine plane by pairs of nonhomogeneous coordinates $Y = (Y_1, Y_2)$, where

$$Y_1 = \frac{y_1}{y_0}, \quad Y_2 = \frac{y_2}{y_0}. \quad (7.17)$$

The allowable representations \mathbb{R}_A of the affine plane are those representations \mathbb{R} of \hat{S}_2 in which the line i has the equation $y_0 = 0$; and this leads at once to the following theorem.

Theorem 7.4. *If \mathbb{R}_A is any allowable representation of the affine plane, then the whole class (\mathbb{R}_A) of allowable representations consists of all those representations, which can be derived from \mathbb{R}_A by applying a transformation of the form*

$$\begin{aligned} Y'_1 &= b_{11}Y_1 + b_{12}Y_2 + C_1, \\ Y'_2 &= b_{21}Y_1 + b_{22}Y_2 + C_2, \end{aligned} \quad (7.18)$$

where the coefficients are arbitrary real numbers subject to the condition $|b_{rs}| \neq 0$.

Using Theorem 7.4, it is possible to transform the n variables of two solutions into a new pair of solutions, according to the following transformation:

$$\begin{pmatrix} Y'_{11} & Y'_{12} & \cdots & Y'_{1n} \\ Y'_{21} & Y'_{22} & \cdots & Y'_{2n} \end{pmatrix} = \begin{bmatrix} b_{11} & b_{12} & C_1 \\ b_{21} & b_{22} & C_2 \\ \text{crossover} & & \text{mutation} \end{bmatrix}_n \begin{pmatrix} Y_{11} & Y_{12} & \cdots & Y_{1n} \\ Y_{21} & Y_{22} & \cdots & Y_{2n} \\ 1 & 1 & \cdots & 1 \end{pmatrix}. \quad (7.19)$$

Equation (7.19) can be expanded to the whole population. The advantages of this representation are as follows.

- (1) Standardized treatment of all transformations.
- (2) Complex transformations are composed from single transformations by means of matrix multiplication.
- (3) An n -dimensional point can be transformed by applying a set of n transformations.
- (4) Simple inversion of the transformation by matrix inversion.
- (5) Extremely fast, hardware-supported matrix operations in high-power graphic workstations.

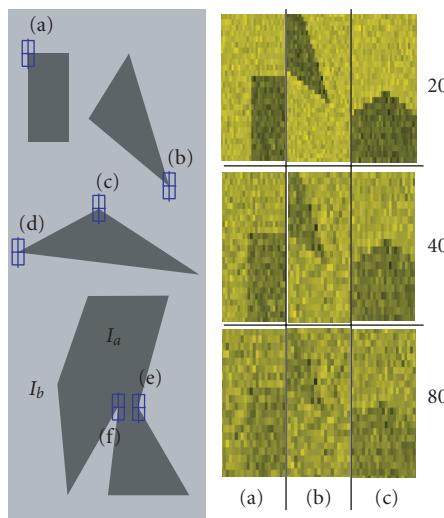


FIGURE 7.6. On the right structures (a), (b), and (c) of the test synthetic image with Gaussian noise scaled by $\lambda = 20, 40, 80$.

7.5. Experimental results

In order to show the robustness of each algorithm and the stability of our L-corner model, we applied Gaussian noise of zero mean and unit variance. This noise is scaled by a constant λ in order to produce perturbations on the synthetic test image. This noise is known as *additive noise* [7]. We decide to use a synthetic image in order to know precisely the location of the corners. The signal-to-noise ratio (SNR) is computed in decibels (dB). Figure 7.6 shows the structures (a), (b) and (c) with the Gaussian noise scaled by the factors $\lambda = 20, 40, 80$. Table 7.1 presents final results of corners (a)–(f) of Figure 7.6, using three optimization strategies without noise. It is important to remember that for $\text{SNR} \rightarrow 1$, the error on the noisy image is approximately equal to the amount of signal of the synthetic image. If $\text{SNR} \leq 1$, the noise is bigger than the original signal, and if $\text{SNR} \gg 1$, the original synthetic image and the synthetic image with noise are equivalents and the error tends to zero. The test of our L-corner detector considering three optimization algorithms was organized as follows.

- (1) The *downhill simplex*, *simulated annealing*, and *evolutionary algorithm* were applied to the structures (a), (b), and (c) of Figure 7.6. Those structures show three different corners: straight-angle corner, acute-angle corner, and obtuse-angle corner, respectively.
- (2) The size of the window is 13×13 pixels, centered around the pixel:
 - (a) $(u_0, v_0) = (87, 87)$;
 - (b) $(u_0, v_0) = (707, 353)$;
 - (c) $(u_0, v_0) = (396, 397)$.

TABLE 7.1. Exact corner location (u_e, v_e) and parameter values \vec{P} using the *downhill simplex*, *simulated annealing*, and *evolutionary algorithm* considering structures (a)–(f) of Figure 7.6.

<i>downhill simplex</i>								
Corner	Initial pixel u_0 v_0		Corner point u_e v_e		Aperture α^o	s_{U_1}	s_{U_2}	χ^2
(a)	89	89	87.404142	87.709506	89.76333	+1	-1	1.13215e - 15
(b)	706	353	707.028574	353.831573	35.7994	-1	+1	5.44007e - 16
(c)	398	397	396.825889	397.466375	149.9239	-1	-1	1.31891e - 09
(d)	44	484	40.985292	484.451324	19.1055	+1	+1	4.22482e + 04
(e)	570	795	570.573303	792.934090	97.4398	+1	+1	1.22339e - 11
(f)	482	795	483.081960	792.272648	37.2723	-1	-1	6.41950e - 23
<i>Simulated annealing</i>								
Corner	Initial pixel u_0 v_0		Corner point u_e v_e		Aperture α^o	s_{U_1}	s_{U_2}	χ^2
(a)	89	89	87.526970	87.437709	90.01414	+1	-1	1.94364e - 10
(b)	706	353	707.017039	353.829675	35.7614	-1	+1	3.56136e - 13
(c)	398	397	396.797404	397.427744	149.4558	-1	-1	9.75333e - 13
(d)	44	484	40.077467	484.514532	16.977	+1	+1	4.22483e + 04
(e)	570	795	570.647257	793.111032	97.7988	+1	+1	6.15267e - 01
(f)	482	795	483.046760	792.255049	37.3776	-1	-1	1.54081e - 13
<i>Evolutionary algorithm</i>								
Corner	Initial pixel u_0 v_0		Corner point u_e v_e		Aperture α^o	s_{U_1}	s_{U_2}	χ^2
(a)	89	89	87.895717	87.644934	89.3094	+1	-1	9.40007e - 06
(b)	706	353	706.807636	353.755519	36.2416	-1	+1	1.24802e - 06
(c)	398	397	396.943119	397.390921	148.7181	-1	-1	5.69696e - 05
(d)	44	484	41.009408	484.473710	18.53073	+1	+1	4.22483e + 04
(e)	570	795	570.411461	792.991919	95.7904	+1	+1	4.53057e - 04
(f)	482	795	483.170778	792.099214	37.0405	-1	-1	1.17246e - 06
(a)	1.24e - 02	1.41e - 02	-1.11e + 00	1.34e + 00	6.73e - 02	-7.57e - 01	103.00	76.00
(b)	4.32e - 03	9.65e - 03	3.20e - 01	-4.52e - 01	-3.30e + 01	-2.07e + 01	103.00	76.00
(c)	9.80e - 03	8.63e - 03	-2.33e + 00	-1.17e - 01	-7.30e + 01	1.43e + 01	103.00	76.00
(d)	3.31e - 02	2.23e - 03	-1.10e + 00	-7.08e - 01	7.58e + 01	-4.43e + 00	102.59	76.45
(e)	1.22e - 03	1.74e - 03	-8.23e - 01	2.32e + 00	3.15e + 01	-3.73e + 01	103.00	76.00
(f)	1.13e - 02	5.92e - 03	4.34e - 01	1.97e + 00	1.43e + 01	3.86e + 01	103.00	76.00

- (3) The control parameters of each algorithm are
 - (a) *downhill simplex*: maximal number of movements of the simplex $N = 4500$;
 - (b) *simulated annealing*: initial temperature $T = 1$, size of the equilibrium state $I = 20$, maximal number of iterations $N = 4500$;
 - (c) *evolutionary algorithm*: crossover percentage $pc = 0.80$, mutation percentage $pm = 0.05$, convergence percentage $pf = 0.75$, offspring number in the population $P = 22$, maximum number of generations $N = 2000$ approximately equivalent to 4500 movements.
- (4) 30 samples for each test were performed.
- (5) Each noisy window $I_r(i, j)$ was normalized to the intensity values [0, 255] considering real numbers through the following function:

$$I_n(j, i) = \frac{255}{\max(I_r(j, i)) - \min(I_r(j, i))} I_r(j, i), \quad (7.20)$$

where $I_n(j, i) \mid i, j = 1, \dots, 2w + 1$ is the normalized studied window including Gaussian noise.

- (6) Threshold is used as stop criterium $f_{tol} = 1 \times 10^{-9}$.

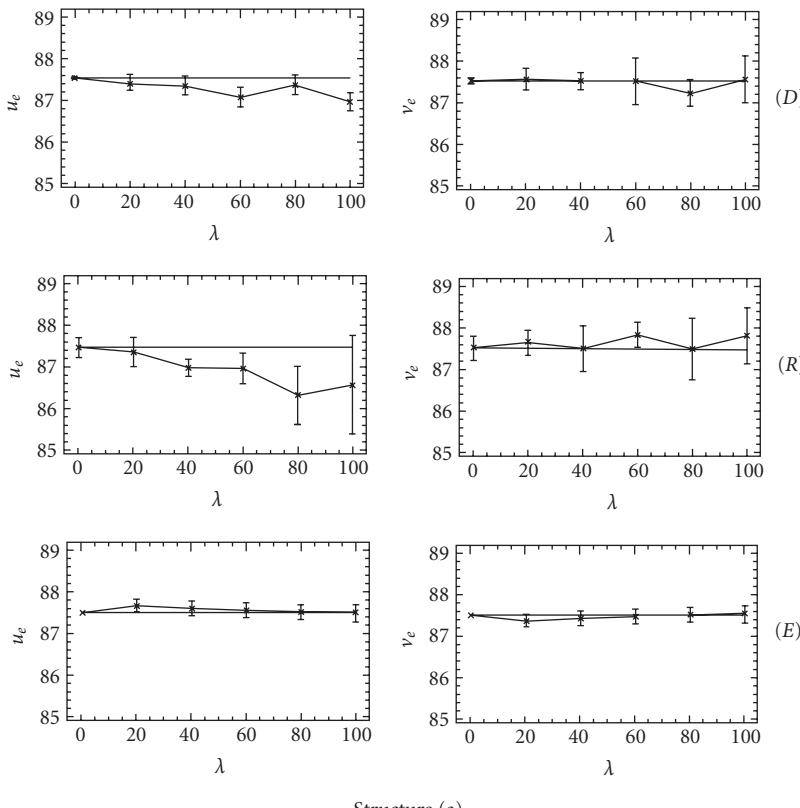
As a result of the test, Figures 7.7, 7.8, 7.9 were built. These figures show the displacement of the corner position (u_e, v_e) of the structures (a), (b), and (c) respectively, considering that a random Gaussian noise was applied over the test image. Each figure shows the average corner point (\bar{u}_e, \bar{v}_e) and its final standard deviation considering 30 samples for each optimization strategy. The charts on the left of each figure represent the u coordinate and those on the right represent the v coordinate of the image coordinate system (u, v) . The horizontal straight line denotes the corner position (u_e, v_e) of the synthetic image with free-noise $\lambda = 0$. After a careful analysis of these figures, we conclude the following.

(1) Beyond $\lambda = 80$ ($\text{SNR} \approx 3.5$), see Figure 7.7, the contours of the structures (a), (b), and (c) are difficult to be distinguished. However, the random Gaussian noise does not blur the borders. Hence, the structure is more or less preserved in shape. Then, it is possible to study the ability that each optimization strategy offers, in order to integrate and reconstruct each structure.

(2) In the case of structure (a), the *evolutionary algorithm* presents the best curve of behavior in presence of noise. Moreover, the maximum standard deviation is obtained for $\lambda = 20$. This value is approximately 0.14 pixels.

(3) In the case of structure (b), the *downhill simplex* presents the best curve of behavior in presence of noise. The maximum standard deviation occurs in $\lambda = 60$ over the u axis. This value is about 0.37 pixels compared to 0.47 pixels obtained by the *evolutionary algorithm* also in $\lambda = 60$. The curve of the *evolutionary algorithm* remains constant around the average value (x_e, y_e) in all cases.

(4) In the case of structure (c), the *evolutionary algorithm* presents the best curve in presence of noise over the u axis. While the *downhill simplex* presents the best curve around the v axis.



Structure (a)

FIGURE 7.7. Behavior of the *downhill simplex* (D), *simulated annealing* (R), and *evolutionary algorithm* (E) considering a Gaussian noise scaled by a factor λ over the structure (a) of the synthetic image.

(5) The following conclusion rises after observing the standard deviation of the algorithms. The *evolutionary algorithm* presents the best average for each experiment as follows: (1) the average standard deviation of the *evolutionary algorithm* is 0.19; (2) the average standard deviation of the *downhill simplex* is 0.25; (3) the average standard deviation of the *simulated annealing* is 0.76 pixels.

As a result, the *evolutionary algorithm* is less sensitive to noise. Hence, it can be considered more robust. However, the *downhill simplex* offers similar results for the L-corner studied here. Finally, the *simulated annealing* shows the worst behavior in the presence of Gaussian noise.

7.6. Summary and conclusions

Accurate L-corner measurement was obtained with a parametric model $M'_L(x, y, \vec{P})$ using a global optimization approach. The goal was to obtain the best set of parameters $\vec{P} = (\sigma_1, \mu_1, \vartheta_1, \sigma_2, \mu_2, \vartheta_2, A, B)$ that fits a window of $2w - 1 \times 2w - 1$ pixels

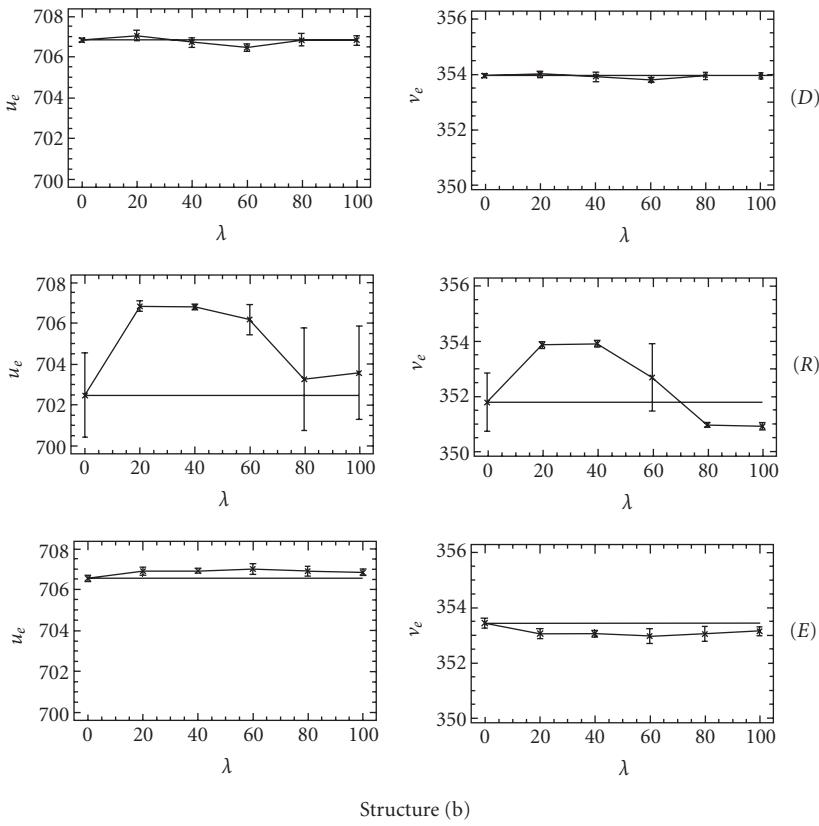


FIGURE 7.8. Behavior of the *downhill simplex* (D), *simulated annealing* (R), and *evolutionary algorithm* (E) considering a Gaussian noise scaled by a factor λ over the structure (b) of the synthetic image.

centered around a pixel (u_0, v_0) within a digital image. The optimization criterion used was the maximum likelihood estimator χ^2 obtained through a multidimensional least squares fitting of the data to the L-corner model. We use the Levenberg Marquardt method, which is a standard technique in computer vision. The Levenberg Marquardt method is considered as a local method. We propose an *evolutionary algorithm* using a novel representation that integrates the two main operators into a single affine transformation. As a result, we obtain a local criterium embedded into a global method. Hence, our algorithms are accelerated by the Levenberg Marquardt Method. Concluding, the *evolutionary algorithm* presents the best behavior against noise. The *downhill simplex* is the simplest strategy to operate because it does not require any control parameter. On the other side, the *evolutionary algorithm* has the ability to increase the population size, which increases the probability to find a better result in a smaller number of generations. The other two strategies do not have this ability. Finally, the *evolutionary algorithm* can be easily parallelized.

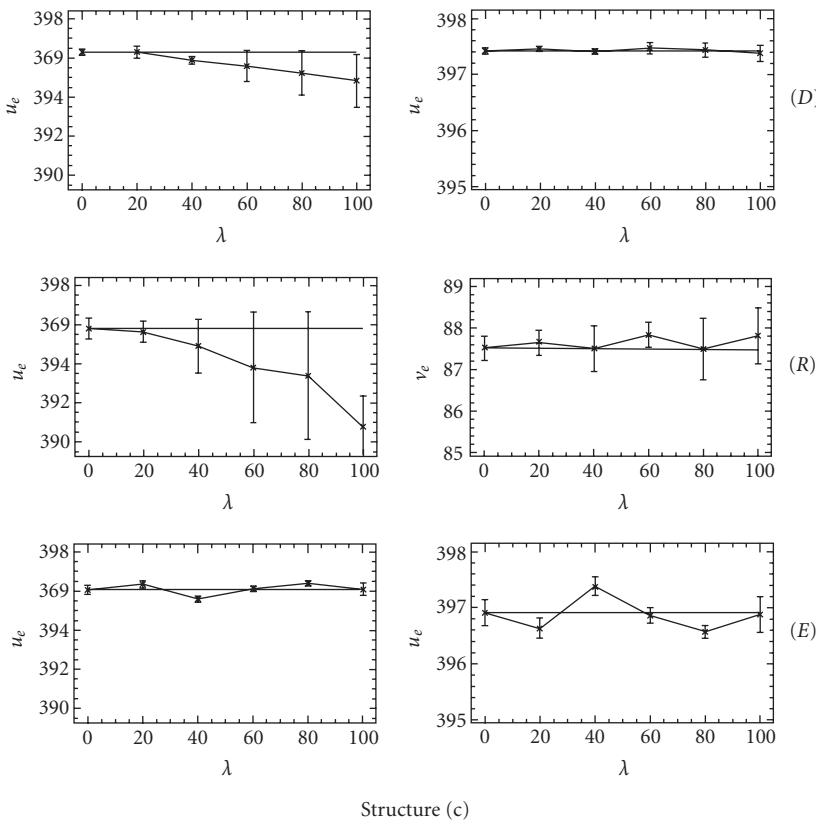


FIGURE 7.9. Behavior of the *downHill simplex* (D), *simulated annealing* (R), and *evolutionary algorithm* (E) considering a Gaussian noise scaled by a factor λ over the structure (c) of the synthetic image.

Acknowledgment

This research was funded through the LAFMI (Laboratoire Franco-Mexicain d’Informatique) project sponsored by CONACyT-INRIA.

Bibliography

- [1] F. Ackermann, “Digital image correlation: performance and potential application in photogrammetry,” *Photogrammetric Record*, vol. 11, no. 64, pp. 429–439, 1984.
- [2] S. Baker, S. K. Nayar, and H. Murase, “Parametric feature detection,” *International Journal of Computer Vision*, vol. 27, no. 1, pp. 27–50, 1998.
- [3] P. R. Beaudet, “Rotationally invariant image operators,” in *Proceedings of the International Conference on Pattern Recognition (ICPR '78)*, pp. 579–583, Kyoto, Japan, November 1978.
- [4] J. Canny, “A computational approach to edge detection,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 8, no. 6, pp. 679–698, 1986.
- [5] R. Deriche, “Using Canny’s criteria to derive a recursively implemented optimal edge detector,” *International Journal of Computer Vision*, vol. 1, no. 2, pp. 167–187, 1987.

- [6] R. Deriche and G. Giraudon, "Computational approach for corner and vertex detection," *International Journal of Computer Vision*, vol. 10, no. 2, pp. 101–124, 1993.
- [7] E. R. Dougherty, *Random Processes for Image and Signal Processing*, SPIE Optical Engineering Press and IEEE Press, Bellingham, Wash, USA, 1999.
- [8] L. S. Dreschner and H. H. Nagel, "On the selection of critical points and local curvature extrema of region boundaries for interframe matching," in *Proceedings of International Conference on Pattern Recognition (ICPR '82)*, pp. 542–544, Munich, Germany, 1982.
- [9] O. D. Faugeras and G. Toscani, "The calibration problem for stereo," in *Proceedings of Computer Vision and Pattern Recognition (CVPR '86)*, pp. 15–20, Miami Beach, Fla, USA, June 1986.
- [10] O. D. Faugeras and G. Toscani, "Camera calibration for 3D computer vision," in *Proceedings of International Workshop on Machine Vision and Machine Intelligence*, pp. 240–247, Tokyo, Japan, February 1987.
- [11] W. Forstner, "A feature based correspondence algorithm for image matching," *International Archives of Photogrammetry and Remote Sensing*, vol. 26, no. 3, pp. 150–166, 1986.
- [12] M. Gen and R. Cheng, *Genetic Algorithms and Engineering Design*, John Wiley & Sons, New York, NY, USA, 1997.
- [13] A. W. Gruen, "Adaptive least squares correlation: a powerful image matching technique," *South African Journal of Photogrammetry, Remote Sensing and Cartography*, vol. 14, no. 3, pp. 175–187, 1985.
- [14] A. W. Gruen and E. P. Baltsavias, "Geometrically constrained multiphoto matching," *Photogrammetric Engineering & Remote Sensing*, vol. 54, no. 5, pp. 633–641, 1988.
- [15] C. Harris and M. Stephens, "A combined corner and edge detector," in *Proceedings of the 4th Alvey Vision Conference*, pp. 147–151, Manchester, UK, August-September 1988.
- [16] L. Kitchen and A. Rosenfeld, "Gray-level corner detection," *Pattern Recognition Letters*, vol. 1, no. 2, pp. 95–102, 1982.
- [17] D. Marr and E. Hildreth, "Theory of edge detection," *Proceedings of the Royal Society of London B*, vol. 207, no. 1167, pp. 187–217, 1980.
- [18] G. Medioni and Y. Yasumoto, "Corner detection and curve representation using cubic B-splines," *Computer Vision, Graphics, and Image Processing*, vol. 39, no. 3, pp. 267–278, 1987.
- [19] R. Mehrotra and S. Nichani, "Corner detection," *Pattern Recognition*, vol. 23, no. 11, pp. 1223–1233, 1990.
- [20] C. Janikow and Z. Michalewicz, "An experimental comparison of binary and floating point representations in genetic algorithms," in *Proceedings of the 4th International Conference on Genetic Algorithms (ICGA '91)*, pp. 31–36, San Diego, Calif, USA, July 1991.
- [21] H. P. Moravec, "Towards automatic visual obstacle avoidance," in *Proceedings of the 5th International Joint Conference on Artificial Intelligence (IJCAI '77)*, p. 584, Cambridge, Mass, USA, August 1977.
- [22] T. Lindeberg, "Feature detection with automatic scale selection," *International Journal of Computer Vision*, vol. 30, no. 2, pp. 79–116, 1998.
- [23] G. Olague and B. Hernández, "A new accurate and flexible model based multi-corner detector for measurement and recognition," *Pattern Recognition Letters*, vol. 26, no. 1, pp. 27–41, 2005.
- [24] G. Olague and B. Hernández, "Flexible model-based multi-corner detector for accurate measurements and recognition," in *Proceedings of International Conference on Pattern Recognition (ICPR '02)*, vol. 2, pp. 578–583, IEEE Computer Society Press, Quebec City, Canada, August 2002.
- [25] G. Olague, B. Hernández, and E. Dunn, "Hybrid evolutionary ridge regression approach for high-accurate corner extraction," in *Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR '03)*, vol. 1, pp. 744–749, Madison, Wis, USA, June 2003.
- [26] G. Olague, "Automated photogrammetric network design using genetic algorithms," *Photogrammetric Engineering & Remote Sensing*, vol. 68, no. 5, pp. 423–431, 2002, paper awarded the "2003 First Honorable Mention for the Talbert Abrams Award", by ASPRS.
- [27] W. H. Press, B. P. Flanery, S. A. Teukolsky, and W. T. Vetterling, *Numerical Recipes in C*, Cambridge University Press, Cambridge, UK, 2nd edition, 1992.
- [28] K. Rohr, "Modelling and identification of characteristic intensity variations," *Image and Vision Computing*, vol. 10, no. 2, pp. 66–76, 1992.

- [29] K. Rohr, "Recognizing corners by fitting parametric models," *International Journal of Computer Vision*, vol. 9, no. 3, pp. 213–230, 1992.
- [30] P. L. Rosin, "Augmenting corner descriptors," *Graphical Models and Image Processing*, vol. 58, no. 3, pp. 286–294, 1996.
- [31] D.-M. Tsai, H.-T. Hou, and H.-J. Su, "Boundary-based corner detection using eigenvalues of covariance matrices," *Pattern Recognition Letters*, vol. 20, no. 1, pp. 31–40, 1999.
- [32] Z. Zheng, H. Wang, and E. K. Teoh, "Analysis of gray level corner detection," *Pattern Recognition Letters*, vol. 20, no. 2, pp. 149–162, 1999.

Gustavo Olague: Applied Physics Division, Computer Science Department, The Center for Scientific Research and Higher Education of Ensenada (CICESE), Ensenada 22860, Mexico

Email: olague@cicese.mx

Benjamín Hernández: Institute of Astronomy, The Autonomous National University of Mexico (UNAM), Michoacán, Morelia 58089, Mexico

Email: benja@astrosen.unam.mx

8

Evolution of an abstract image representation by a population of feature detectors

Leonardo Bocchi

8.1. Introduction

Image segmentation is an essential step toward image understanding because it allows to decompose the image into a set of units (or regions) representing the actual objects which can be observed in it. In computer vision, segmentation is the step of the elaboration chain where we start to analyze the logical and structural relations between the entities, or the features, which are present in the image and that have been emphasized during the low-level processing steps.

In the simplest case, to segment a (static and nontextured) image means to identify regions composed of pixels having similar grey levels, and, therefore, the boundaries between those regions. In the general case, we are interested in the detection of regions composed of pixels having a homogeneous value of some property, which can be related to texture, color, motion, or whatever else, depending on the actual application.

Region detection can be achieved using different approaches. For instance, one may start the process by identifying the edges between objects and grouping them to form boundaries between regions. In this case, we obtain a boundary-based segmentation. In an opposite way, a region-based segmentation may be achieved by grouping together pixels having similar properties. Both methods may be described using a graph representation, where a set of nodes, associated with the regions detected in the image, are connected by arcs describing the adjacency relationships between regions. In a similar way, the same graph may be realized using a set of arcs, representing boundaries between regions. The arcs connect nodes, which represent the points where more than two regions meet.

In both cases, the method is completely data-driven: the outcome of the segmentation process is determined only by the features (edges or gray levels) present in the image, and the method does not make use of any prior knowledge about the type of objects that are present in the image.

A feasible alternative, which allows to introduce stricter constraints on the kind of regions, or objects, which are identified by the segmentation process, is

represented by deformable models. The expected shape of the object to be localized is embedded into a set of primitives describing the model. The segmentation is carried out by an optimization process, which involves minimization of an objective function depending both on the matching between the deformed model and the observed image and on the degree of the deformation itself.

Several deformable models have been proposed in the literature, starting from snakes [15]. A snake, or active contour, represents the boundary of an object, which is modeled as an elastic band. The band, according to the elastic model, has an internal energy given by bending and stretching. Therefore, we can embed knowledge of the desired shape in the elastic parameters which describe the rigidity of the band. While the original snake algorithm involved variational calculus, a large number of minimization techniques have been proposed. Those include, for instance, simulated annealing [13, 22], dynamic programming [1, 10] and greedy algorithms [14, 25].

Evolutionary and genetic methods have been extensively proposed in this field, both to identify the optimal edge detector and elastic model parameters [5] and to carry out the minimization process [3].

Snakes are an example of the so-called free-form elastic models, which do not impose a strong constraint on the shapes of the object. When some prior knowledge of the geometrical shape is available, it is possible to incorporate this knowledge in the elastic model. In this way, it is possible to constrain the shape of the stretched model to be compatible with the object to be detected. At the same time, the introduction of constraints on the model shape reduces the number of free parameters which are needed to describe the model, increasing the efficiency of the method. In the approach known as parametric approach, the elastic model is composed of a prototype template and of a parametric mapping. The template may be represented as a mean shape of the expected object, computed from a learning set. The parametric mapping usually describes the most *natural* deformations of the shape. Several authors proposed methods to extract informations about the deformations from a training set. For instance, Grenander and Miller [11] described a systematic framework to deal with shape deformations, while Cootes et al. [6] associate the deformation patterns to the eigenvectors of the covariance matrix of deviations from the mean shape.

The drawback of the parametric approach is the complexity of the procedure which is necessary to model the shape of the object and its deformation patterns. Moreover, most of the proposed methods require an accurate manual segmentation of a large image dataset in order to extract the necessary information.

A possible alternative approach, to avoid the overhead of a training phase requiring manual intervention, is to use a self-organizing system, as the one which can be achieved through neural networks [17]. Indeed, neural architectures have been extensively proposed to identify correspondences between similar images which have the same overall structure, but exhibit small differences which cannot be represented by linear geometric deformations. Matching corresponding points of different images is a problem which arises in several fields of the computer vision

theory. For instance, this problem arises when we try to reconstruct 3D information from stereo images: the key point is the identification of matching pairs of points in two projections of the object [2]. Other common examples are object recognition [4] or tracking.

An interesting neural architecture which has been proposed in order to perform a matching between images is the dynamic link architecture (DLA) [18, 24, 26]. DLA is a neural architecture able to associate each image with a graph. Image recognition is based on the best match between one of the stored graphs and the unknown image.

Each image is represented using a set of Gabor jets and the matching process is done by minimizing a cost function which takes into account both graph distortion and similarity between sets of Gabor features. The input image is then associated to the image corresponding to the best matching graph.

A possible extension is to build an architecture, somehow similar to DLA, but with the added feature of being able to create its own knowledge representation in an unsupervised way. This architecture will, therefore, build an abstract representation of the image domain, which includes several representations of the same class of objects. In order to achieve this result, the system needs to reveal the common features which appear in all, or in most, input images. It also needs a method to identify and extract the spatial relationships between the detected features.

An architecture which is able to extract knowledge from a set of unlabeled examples is the self-organizing map [17], which has several points of contact with the DLA. The SOM, however, is designed to classify patterns which are independent from each other. In the image segmentation task, patterns (corresponding to local features of the image) have a set of spatial relationships coupling neighboring patterns. Actually, a large part of the information about image content is not embedded in the pattern itself but in the relationships between patterns, that is, the histogram does not describe the image because it takes only into account the pattern (in this case, the gray level) and does not involve the relationships among patterns (the spatial distribution of gray levels).

In order to capture the relevant part of the information, it is necessary to change the concept of “organizing” in the SOM definition. The meaning of the word “organization” in the SOM relates to similarity between patterns: similar patterns are mapped onto neighboring units. In order to capture the spatial organization, it is necessary to map patterns, which are spatially related, into neighboring units.

A possible way to obtain this behavior is to transform neural units, which are statically linked to a lattice, into active particles, each of which is adapted to a particular feature of the image, which can move on the image plane.

When a new image is presented to the system, each particle begins to “chase” local features of the image, trying to reach points where features appear. In order to maintain spatial relationships between particles, an attraction among neighboring particles has been introduced. Due to the introduction of this force, when compared to a standard particle swarm optimization (PSO) [8, 16], the particles

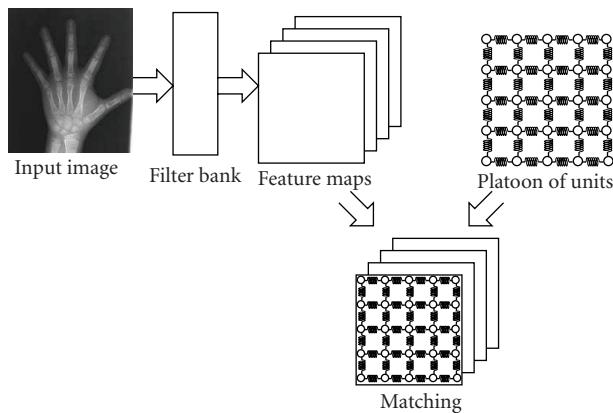


FIGURE 8.1. System architecture.

are constrained by the attraction forces to move in a much more coordinated way, acting more like a platoon of soldiers than like a swarm of independent particles.

The resulting architecture, called elastic neural matching (ENM), can be described as lattice of active neural units which is stretched until each particle (representing a local feature) localizes the matching feature in the sample image. The stretching process builds a mapping from the abstract representation stored inside the unit weights into the image plane.

The performances of the proposed ENM network have been evaluated on the problem of automatic labeling of anatomical regions in a set of hand radiograms.

8.2. System architecture

The overall system architecture may be schematized as a feature extraction block feeding a platoon of active neural particles, as shown in Figure 8.1. The feature extraction block processes an input image and produces a vector of feature maps. Each feature map represents the spatial distribution of some local feature of the image, and associates to each pixel of the input sample a feature vector describing the local properties of the image in that point. The image may be described as a world where particles live. Each location, or pixel, of this world is associated with a local environment which is represented by the feature vector calculated from the local values of the feature maps.

The neural network is composed of a platoon of neural particles, which at rest are arranged on a square lattice. Neighboring particles are attracted toward one another by “friendship” which is modeled as an elastic force having an elastic constant c . The particles are assumed to differentiate during system adaptation, so each neural unit u_{ij} is associated to a personal ecosystem, modeled as a vector of weights w_{ij} , which represents its ideal environment, or the environment which is best suited for the particle.

In order to analyze an image, the platoon is superimposed on the vector of feature maps. In this way, each unit u_{ij} is located in a position $P_{ij} = (x, y)$ in the image plane. The position P_{ij} is called *match point*. Each unit is an autonomous entity, which is set free to move around and look for a good environment for living. In this process, the particle acts as a specialized feature detector, which tries to locate, in a neighborhood of the match point, the point in the image having a feature vector matching, as close as possible, the personal ecosystem of the unit. In other words, the particle looks for a position, in a local neighborhood, which has a local environment as close as possible to the ideal environment for the particle.

The unit is therefore attracted toward this position and starts to move in this direction, while elastic constraints expressed by the elastic forces constrain neighboring units to move in a coordinated way.

Without neighborhood attraction, the behavior of particles is similar to the behavior of a group of roaches dropped on the floor: each roach starts to run toward the best place for hiding, according to its judgement (in the model terminology, each roach looks for the place which best matches its ideal environment). The attraction forces can be represented, in this example, as a set of springs connecting together roaches and forcing them to move in a coordinated way.

It is possible to describe the attraction force and the elastic bindings in terms of energy of the unit, while the movement is associated to a process of energy minimization, which will be called relaxation.

When the process is completed, each unit is located on the point in the image which best corresponds to the personal environment stored in the vector of weights in the unit and which best fulfills the topological relationships between neighboring units. When this phase is completed, each particle may adapt to its location, increasing the correspondence between the local environment and its ideal environment.

Iterating this process over different images, the relaxation step and the following adaptation of the weight vector let the network units differentiate from one another, and each of them adapts to live in an environment described by a certain feature vector, which appears in the input images, also when such feature vector is located in different positions in the input samples. A particle which is often located in a certain environment (e.g., a vertical edge) gets used to living on vertical edges, wherever they appear in the input. When an unknown image is presented to the system, the particle tries to reach a position which reminds it of a vertical edge. At the beginning of the relaxation process, the unit tends, therefore, to move toward any edge having the same direction. In the same time, attraction forces which act on the units keep them from moving freely on the image. This forces neighboring units to adapt to identify feature vectors which occur in neighboring points in the input images used during the training phase. In this way, we obtain a mapping between feature vectors and weight vectors, having the property that feature vectors which are spatially adjacent in the image (although having different values) are mapped into units spatially adjacent in the network grid.

For instance, Figure 8.2 shows a column of units (a), adapted to live on a vertical edge. A curved edge is then presented to this part of the network (b). The

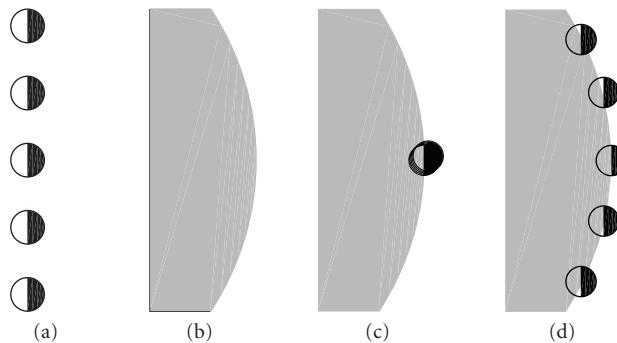


FIGURE 8.2. Example of network relaxation. (a) Five units have adapted to identify a vertical edge. (b) A curved edge is presented. (c) The best matching between units and image: all units are located on the vertical part of edge. (d) Elastic constraints force units to space almost evenly.

optimal match between the units and the image occurs in the point of the edge which has vertical direction, and all units are attracted to reach that point (c). Using a different example, if different kinds of food are placed on a surface, a swarm of units without constraints group together in the position where the most attractive spot of food is located. The elastic constraints, however, prevent an excessive crowding of units by maintaining their relative distances. The balance between the two forces distributes the units as in (d), where the elastic force due to stretching balances the attraction toward the point where the edge has a vertical direction.

8.3. Feature maps

A proper selection of the set of feature maps is of primary importance to obtain good performances of the system. As the matching between units and points of the image is done by comparing the unit weights with the vector of features in that point, it is mandatory that the features represent the local properties of the image (presence and directions of edges, corners, and similar characteristics of the image). Moreover, a set of optimal features needs to provide a compact description of these local properties of the image, to have a feature vector of reasonable size. This suggests that a feature set be selected having both a limited spatial support, in order to capture local information, and a limited frequency response allowing to reduce noise. These properties, as shown by several researchers [7, 19], are best exploited by Gabor functions:

$$\gamma_k = \alpha_{k,\sigma} \exp \left(-\frac{x^2}{2\sigma^2} \right) \exp(ikx), \quad (8.1)$$

where the constant $\alpha_{k,\sigma}$ is a normalization constant to ensure that $\|\gamma_k\| = 1$. By varying the parameters describing this basic function, it is possible to obtain several families of curves, commonly called wavelets.

We used a family of functions, called Morlet wavelets, strictly related with (8.1), but with the additional properties of being self-similar at different scales. Morlet wavelets, in the monodimensional space, can be expressed as

$$\psi_k = \beta_{k,\sigma} \exp \left(-\frac{k^2 x^2}{2\sigma^2} \right) \exp(ikx). \quad (8.2)$$

The extension of (8.2) to bidimensional images is achieved by replacing k and x with vectors, and interpreting the products as scalar products:

$$\psi_{k,\theta} = \beta_{k,\sigma} \exp \left(-\frac{\mathbf{v}_k^2 \mathbf{x}^2}{2\sigma^2} \right) \exp(i\mathbf{v}_k \cdot \mathbf{x}), \quad (8.3)$$

where $\beta_{k,\sigma}$ is a normalization constant. The parameter \mathbf{v}_k identifies the single Morlet function, determining its scale and orientation: the scale k of the wavelet, expressed in half octaves, is related to \mathbf{v}_k by the relation $|\mathbf{v}_k| = 2^{k/2}$ and the direction θ of \mathbf{v}_k gives the orientation of the function.

Morlet wavelets can be used to define the (discrete) Morlet transform. Starting from (8.3), the (discrete) Morlet transform is defined as

$$M_{k,\mathbf{x}_0} = \sum_{\mathbf{x}} \psi_{k,\theta}(\mathbf{x} - \mathbf{x}_0) \cdot I(\mathbf{x}). \quad (8.4)$$

For each value of k and θ , (8.4) can be interpreted as a map of features, having a given scale and orientation, which are present in the original image.

8.4. Relaxation

When an image (and its associated feature maps) needs to be presented to the network inputs, the position of each unit is reset to a start value \mathbf{P}_0 . These points are arranged on an equally spaced square grid over the input image. When the units are superimposed to the image, one can expect that the input weights of each unit do not match the features in \mathbf{P}_0 . In other words, each unit will match a point in the image which is located at a certain distance from the start point. Therefore, it is necessary to introduce a relaxation phase, which consists of a series of operations which will allow each particle to move toward the point in the image which best suits its ideal environment.

8.4.1. Network energy

The dynamic behavior of the swarm of particles may be simulated either using differential equations, or through an energy minimization problem. The energy minimization problem has been preferred due to the great compactness of the equation set, and the greater rejection of local noise.

It is possible to introduce a system energy as the sum of an internal energy E_i , corresponding to the elastic energy of the attraction forces, and an external

energy E_e which describes the match between each unit and the environment in the same position of the image. Assuming that each unit is connected to the units of a 4-connected neighborhood by elastic springs having an elastic constant c , the corresponding elastic energy is expressed as

$$E_i = \sum_{(i,j)} \sum_{(m,n) \in S_{ij}} c |P_{ij} - P_{mn}|^2, \quad (8.5)$$

where S_{ij} is the 4-connected neighborhood of the unit u_{ij} and (i, j) are varied over the whole network grid. As for the external energy, it is possible to introduce several expressions to describe the similarity between the two vectors. Among those, the most straightforward selection is obtained using the scalar product

$$E_e = - \sum_{(i,j)} \mathbf{w}_{ij} \times \mathbf{I}(P_{ij}), \quad (8.6)$$

where $\mathbf{I}(P_{ij})$ is the feature vector in the point P_{ij} of the image plane. The total energy of the mesh is expressed as

$$E_t = E_i + E_e, \quad (8.7)$$

where the relative importance of the two components E_i and E_e can be balanced by varying the elastic constant c . Lower values of c allow for a larger deformation of the mesh, while higher values of c prevent the mesh from stretching.

The total value of the energy E_t represents a sort of fitness of the entire population of units. The population has two ways to improve fitness: a short-term relocation and a long-term adaptation. The short-term relocation, associated to the energy minimization during the relaxation phase, from an evolutionary point of view corresponds to finding the best distribution, over the available space, of the individuals belonging to the population, without any evolution of the individuals. During the long-term adaptation, each individual evolves trying to increase its fitness, measured as the correspondence between its personal ecosystem and the local environment.

8.4.2. Attention focus

The relaxation process is performed iteratively. In each step, we randomly select an attention focus, corresponding to an image point P_s . The attention focus is shown as a white circle in Figure 8.3(a). Because different points on the image provide different amounts of information about the image structure, we assume that a uniform probability is not the best distribution to use. Therefore, we developed a spinning wheel method which increases the selection probability of significant

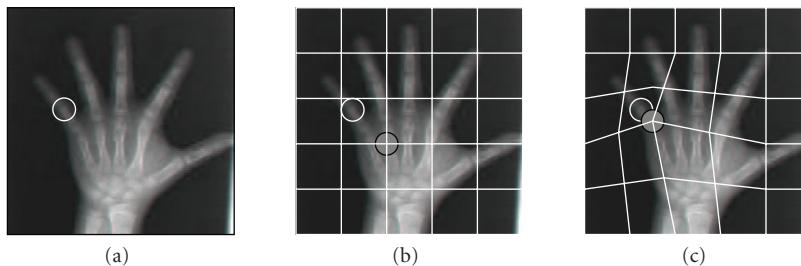


FIGURE 8.3. Relaxation process: (a) selection of the attention focus, (b) detection of the best-matching unit, (c) stretching of the network lattice.

points (e.g., edge points) with respect to points contained within regions having a constant grey level. The resulting point selection scheme has performances similar to the human vision system, where the fixation points of the gaze accumulate on edges of the observed scene, disregarding zone with uniform color, as they have usually a low information content.

The proposed spinning wheel procedure starts by assigning to each point in the image a segment of the wheel proportional to its significance, which is obtained as the sum of squares of the corresponding pixel in the feature maps. The definition of the feature maps we used gives edge points a higher probability of being selected for evaluation. Afterwards, a random selection of a point of the wheel, with uniform probability, allows to select points of the image based on the desired probability distribution.

After the selection step is completed, the selected point, called P_s , acts as an attention focus, which stimulates the particles which are located in a local neighborhood and causing the activation of one of them. In order to get attracted by the attention focus, and therefore to get activated, a particle needs to detect that the attention focus presents a local environment which is better suited to it than the local environment of the point where it is located (decrease of external energy), but taking into account the attraction which ties the particle to its neighbors, preventing it from getting attracted by locations which are too far from its current position (internal energy).

In order to determine which unit gets activated, each unit in a suitable neighborhood is tested by moving its match point from its current position P_{ij} to P_s , and evaluating the decrease of the mesh energy before and after the unit is moved. The active unit, shown in Figure 8.3(b) as a gray-filled circle, corresponds to the unit which produces the maximum decrease of the energy.

8.4.3. Lattice deformation

Once the activation process is completed, the activated unit (i, j) moves toward the sample point P_s of a distance $\lambda(P_s - P_{ij})$ (Figure 8.3(c)). The parameter λ , with $0 < \lambda < 1$, represents a speed factor for the units. With $\lambda = 0$, units cannot move, while $\lambda = 1$ allows the winning unit to move from P_{ij} to P_s in one step. Together with

the winning unit, to maintain the spatial organization of the particles, neighboring units P_{mn} , included in a circular domain S , are also moved in the same direction, according to the law

$$\Delta P_{mn} = \lambda (P_s - P_{mn}) \left(1 - \frac{|(i, j) - (m, n)|^2}{r_{\max}^2} \right), \quad (8.8)$$

where r_{\max} is the radius of the domain S .

After the winning unit has moved, the process continues iteratively by selecting a new attention focus. The process is stopped either when a prefixed number of iterations has been performed, or when the units reach a stable configuration, and differences $P_s - P_{ij}$ are small enough.

8.4.4. Parameter selection

The number of iterations which are required to reach an equilibrium point depends greatly on the parameters used in the training process: the elastic constant c , the relaxation speed λ , and the radius r_{\max} of the domain S . A high value of c , increasing the stiffness of the lattice, prevents units from being activated when the attention focus is too far from the current position of the unit, slowing down the deformation process. On the other end, a low value of c may reduce the stabilizing effect of the grid, allowing the network to lose the spatial relationships between neighboring units. The selection of a proper value of λ , too, greatly influences the convergence speed of the network, as a high value of λ dramatically improves the speed, but increases the chances of instability of the final configuration, where units may “bounce” between two attraction points. The stability of the process, however, may be increased using a high value of r_{\max} . Indeed, this parameter, forcing neighboring units to move in a coordinated manner, helps both to preserve spatial relationships and to avoid the temporal instability. On the other hand, as units are not allowed to move independently from one another, it decreases the precision of the network: the distance between the final position of the unit and the point which best matches its feature vector tends, on average, to increase.

A commonly used solution (which was introduced in the Kohonen map) adopts a set of parameters which are slowly varied during the iterative process. Indeed, in the first iterations, a large value of λ and a small value of c produce a fast deformation of the network where the units move close to points which match their features well, while a large value of r_{\max} prevents a loss of organization in the network. Later, when the network is already partially deformed, a reduction of λ and r_{\max} helps the units to properly position themselves in the best-matching point, while the increase of c prevents further deformations of the lattice. Therefore, it has been assumed that a set of linear expressions can be used to

describe the variation of the relaxation parameters:

$$\begin{aligned}\lambda &= \left(1 - \frac{i}{i_{\max}}\right)\lambda_0, \\ r_{\max} &= \left(1 - \frac{i}{i_{\max}}\right)r_0, \\ c &= \left(1 - \frac{i}{i_{\max}}\right)c_0 + \frac{i}{i_{\max}}c_f,\end{aligned}\quad (8.9)$$

where i is the iteration number, i_{\max} is the maximum number of iterations, λ_0 , r_0 , and c_0 are the initial values of λ , r_{\max} , and c , while c_f is the final value of c .

8.5. Adaptation

The network, in order to build a knowledge about the imaged objects, needs to be trained with a set of images which represent the same class of objects. During the training process, each particle differentiates from the others, progressively adapting to live in a particular environment. The training procedure is similar, apart from the relaxation step, to the training of an unsupervised neural network, which can be described as follows. After a suitable initialization of the weights, the patterns, or images, in the training set are presented to the network inputs, usually in a random order. For each input pattern, the output of the network is evaluated, and some, or all, of the weights of the network are adapted. The process is then iteratively repeated until convergence.

Once the training process has been completed, the weight of the units embed an abstract representation of the imaged objects. Indeed, each unit represents a feature which is present in the images, while topological relations between features in the image are translated into the topological relation between the units representing them. A simple example of the obtained representation is described in Figure 8.4. The figure represents a small ENM network trained with a set of digitized hand-drawn capital H (top row). In this case, the selected feature maps include the gray level of the image, and a lowpass filtered version of the same image. After the training, the first component (which corresponds to the gray-level feature map) of the unit weights contains a “mean” shape of the letters used during the training (bottom row). The network has stored in its weights an abstract representation of the image set, by discovering that all the input images contain the same basic shape, although stretched and deformed.

The adaptation process is performed iteratively. After an initialization phase, an image from the data set is presented to the system, which relaxes, allowing units to find a suitable position in the image. Then, each unit adapts to live in that position, by increasing the match between its personal ecosystem and the local environment present in that point. At this point, it is possible to present a new image and repeat the process until the system reaches the desired adaptation level.

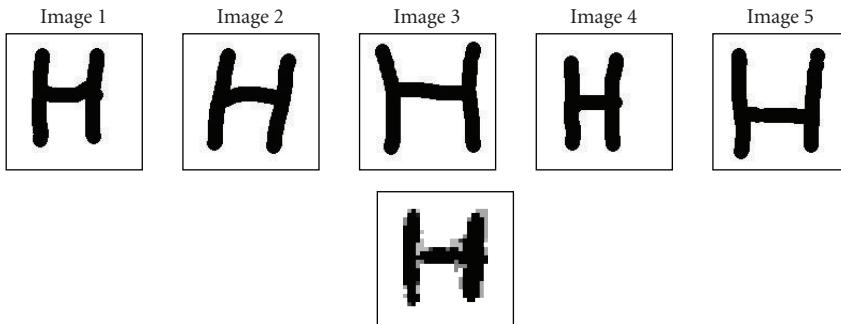


FIGURE 8.4. Abstract representation extracted by a network trained with five images of a capital H (top row). The network weights (bottom row) shown that the network has detected that a single shape is present in all the images.

8.5.1. Initialization

The initialization of the network weights can be done either by a random selection of the weights in a suitable range of values, or by direct assignment to some known value. To enhance stability and convergence speed, it is usually advisable to assign some “reasonable” value to the weights of the units. In this case, a good initialization value can be obtained by sampling the images in the training set. The network grid may be used to assign to each unit a weight vector equal to the feature vector evaluated in the match point before relaxation.

8.5.2. Presentation of input patterns

The main loop of the training phase starts by presenting the input patterns, one at a time, to the network. In order to improve the learning and the generalization properties of the network, the order of presentation of the input patterns is randomized before each iteration. When each pattern has been presented to the network, the relaxation process starts as described above. After the relaxation process has completed, it is possible to let the units adapt to the new environment by changing the associated weight vector. It is worth noting that in this kind of architecture, the activation of the units does not relate to the output of the network. The output, indeed, is described by the final position of each unit in the image plane. For instance, Figure 8.5 reports the deformation patterns of the network grid caused by the inputs reported in Figure 8.4 to the trained network.

8.5.3. Learning rule

The training phase, after the relaxation process has completed, has been designed to increase the fitness of the unit to the environment, by increasing the match

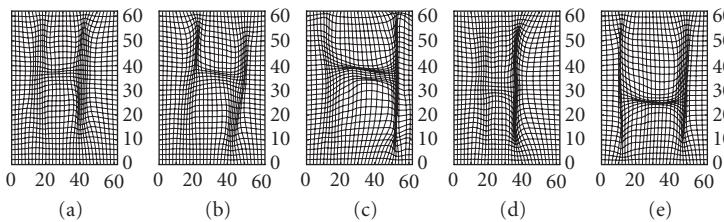


FIGURE 8.5. Deformation of the network grid representing the mapping between input images and network units for the data reported in Figure 8.4.

between the weights of each unit and the image features. This has been done by modifying the weights of the unit according to the following rule:

$$\Delta \mathbf{w}_{ij} = \epsilon (\mathbf{I}(\mathbf{P}_{ij}) - \mathbf{w}_{ij}), \quad (8.10)$$

where ϵ is the learning ratio, bounded between 0 and 1, and P_{ij} is the position of the unit at the end of the relaxation process.

The proposed learning rule, similar to the one introduced by Kohonen [17], improves the fitness of the unit by increasing the correspondence between each unit and the image features. In the following relaxation steps, the unit will be, therefore, attracted toward a point with the same feature vector. In this way, the unit learns to act as a feature detector specialized to identify points having a feature vector similar to $\mathbf{I}(\mathbf{P}_{ij})$.

8.5.4. Iteration

The process described above for a single image needs to be repeated for all the images in the training set, which are presented to the network in a random order. The entire sequence is then repeated iteratively, for a fixed number of iterations. As happens for the parameters which drive the relaxation process, the best results can be obtained by using a linearly decreasing learning rate.

8.6. Medical application

The system has been evaluated on a set of hand radiograms, in order to identify each bone on the radiogram. Once the bones have been detected, they can be classified in order to obtain an assessment of the maturation level of the skeleton and of the bone age.

8.6.1. Bone age assessment

The assessment of skeletal age maturity is a commonly used procedure in pediatric radiology. A discrepancy between bone maturity and biological age indicates the presence of some abnormality in skeletal growth. The assessment is commonly performed using a radiogram of the left hand, due to the relatively small patient

exposure to radiation and the simplicity of the test. Moreover, the hand presents a large number of ossification centers which can be used to obtain an accurate estimation of the degree of maturity.

Several methods have been proposed to allow the medical expert to obtain a quantitative estimation of the skeletal age. Among them, the most commonly used procedure is the Greulich and Pyle atlas matching method [12]. This procedure is based on an overall matching between the observed radiogram and a set of reference images grouped in an atlas. The assessment consists of identifying the image in the atlas which most resembles the examined radiogram.

The selection of the correct image is very difficult in cases where the radiogram presents a growth abnormality which affects the various bones in the hand with a different degree of severity. In this situation, the outcome of the test depends on the relative importance given to different bones by the observer.

The major drawback of the atlas matching method is the high degree of intra- and interobserver variability due to the subjective comparison of the images. A method which reduces the variability of the estimate is the Tanner and Whitehouse method (TW2) [23].

The TW2 method uses a detailed analysis of twenty bones in the hand, which are depicted in Figure 8.6. Each complex in the set is assigned to one of nine maturation classes, labeled from A (no calcified bone is present) to I (maturation completed). This method produces a description of each bone in terms of scores. The sum of all scores assesses the bone age. This method has a higher repeatability, but the required execution time and its complexity prevent its rate of application from exceeding 20% [21].

8.6.2. Material and methods

A successful application of the TW2 method requires a precise localization of the bone complexes which have to be analyzed. Localization and labeling of the bones is still the most challenging step toward the complete automation of the process.

Several factors concur to make identification a complex task. First of all, bones vary greatly during the growth process. In early stages, some bones are either absent or very small, and consequently they produce well-separated shapes on the image. During growth, carpal bones tend to enter in contact and overlap with one another (cf. Figure 8.6 with the sample image in Figure 8.7, where the bones in the carpal region are overlapping). In contrast, epiphyseal and metaphyseal bones start to join, until they form a single bone.

In the meantime, the overall size of the hand and its mean gray level vary greatly. One more source of problems is given by the hand position in the image. Even if great care is applied during the imaging process to correctly position the hand in the radiographic machine, the angles between fingers have a high degree of variability.

Many authors proposed different strategies to overcome this problem and to identify the location of bones in the fingers. Niemeijer et al. [20] propose a method to detect a single bone region. They use an active shape model to align the sample

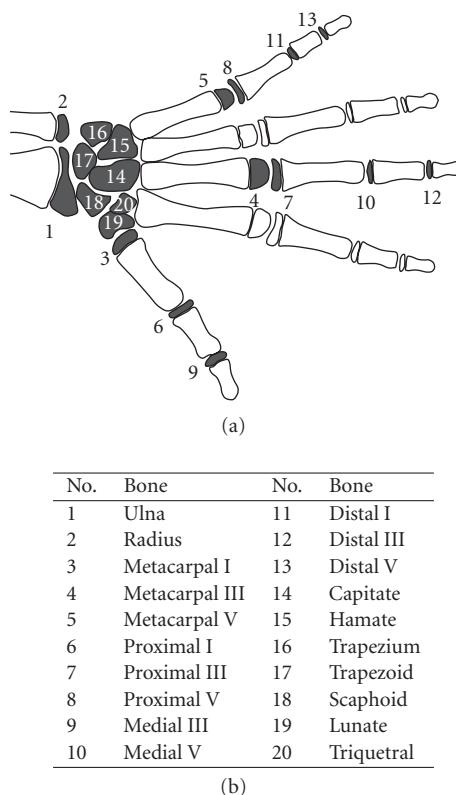


FIGURE 8.6. Bone complex used in TW2 method (1–13: epiphyseal bones, 14–20: carpal bones).

image with a template of the desired bone. A few others tried to localize carpal bones. For instance, Fan et al. [9] adopt a two-stage edge detection method for the detection of a set of reference points. Using the reference point information, they can locate the carpal bones.

8.6.3. Network configuration

An elastic neural network has been tested on a set of radiographic images of the left hand. The radiograms have been acquired using an optical scanner, with a spatial resolution of 100 dpi and a pixel depth of 12 bits, which means the image size is in the range from 500×700 pixel to 800×1000 pixel. The images have been labeled by a radiologist which has identified the ossification centers to be used for the application of the TW2 method. These points have been used as a reference to evaluate the performance of the network. In other words, the identification done by the expert radiologist has been assumed to be absolutely correct. The data set has been used to form a training set, composed of 20 radiograms, and a test set, also containing 20 images.

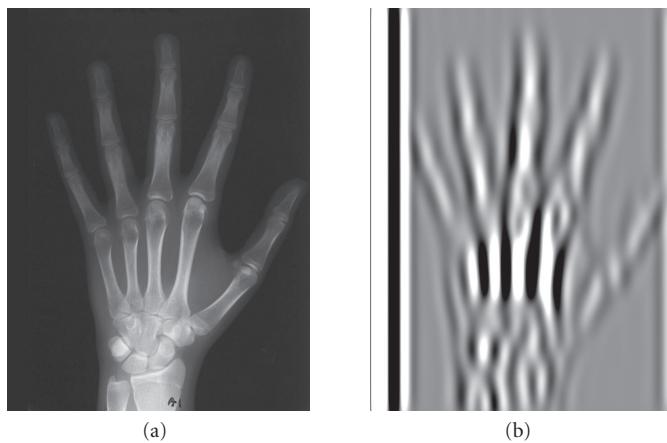


FIGURE 8.7. A sample image from the dataset (a) and one of its feature maps (b).

Given the average image size, it is possible to select the number of units which belong to the neural network, by defining the average distance between two neighboring units. In this case, this distance has been assumed to be around 10 pixel, and therefore the number of units on each side of the grid is about 60, which has been rounded down to 50.

The selection of the desired feature maps has been done using the same heuristic approach. When a small scale is selected, the feature maps contain a lot of fine-detail features which can hide the overall structure of the image, while the selection of a scale which is too large may smooth out important properties of the image. According to this principles, in this application example we selected a set of feature maps having the same value of $|k| = 9$ and four different orientations, corresponding to angles $\theta = 0, \pi/4, \pi/2$, and $3\pi/4$. This selection gives a single-scale representation of the image.

The selection of a scale $k = 9$ allows to smooth out part of the fine structure of the bone. Lower values of k allow for the presence of high-frequency details which could prevent the network from identifying the global minimum of the energy, corresponding to the correct deformation of the network, by trapping units on local minima. On the other side, higher values of k will decrease the accuracy of the matching performed by the network. Figure 8.7 shows a sample image from the data set and one of the corresponding feature maps.

8.6.4. Network training

The network has been trained using 100 iterations over the data set. During each iteration, all images have been presented to the network in a randomly selected order.

Several combinations of network parameters have been tested in order to compare the relative influence on the learning results. A good combination of parameters can be obtained by keeping reasonably low the relaxation speed (around

0.2-0.3) and high the number of relaxation steps (around 10^6 , which means that each unit, on the average, gets activated a few hundred times). Moreover, relaxation speed has been reduced linearly during the process.

The results obtained during the training phase indicate that a larger size of the neighborhood S increases the robustness of the network against local minima, and helps the units to maintain their relative positions. At the same time, however, a smaller neighborhood increases the accuracy of the final positioning of the units. In the final configuration, we used a neighborhood size starting with a radius equal to 8, which linearly decreases during the relaxation phase.

As concerns the learning ratio, it can be observed that it strongly affects network convergence time. Best results are obtained when the learning ratio is decreased during the training process. It conforms to the results achieved in most self-organizing architectures [27].

8.6.5. Evaluation of results

Once the training phase was completed, the results were evaluated by feeding the network with a set of labeled images, where a set of tag points had been identified. The procedure has been designed to transfer the labels from an image pixel to network units, and to transfer them to a different image. It can be described as follows (Figure 8.8). Each test image is applied to the network input, and the mesh completes the relaxation process. At the end of the relaxation process, it is possible to identify the unit whose match point is located closest to the label placed on the image. That unit is assumed to represent the tag point in the first image. When a new image is presented to the network and the relaxation process has completed, the unit is located in a new point. This point is assumed to represent the label in the new image.

An evaluation of the performance of the network can be therefore achieved by measuring the distance from the transferred label and the manually applied label. A perfect matching between the images would produce a null relative displacement, as the same unit should be located on the same marker in all images. Experimental results indicate that the average absolute displacement between the marks is about 1.5 units.

The average execution time required to complete the relaxation step is between 2 and 5 minutes, on a 1.6 GHz PC.

8.7. Conclusions

The proposed architecture is designed to learn the common image structure embedded in a set of images representing the same class of objects. During the training phase, the image structure is stored in the unit weights, which may be viewed as a representation of the topological relationships between image features.

The results indicate that the proposed architecture can be effectively used to recognize complex images, containing several structures, as a hand radiogram. It

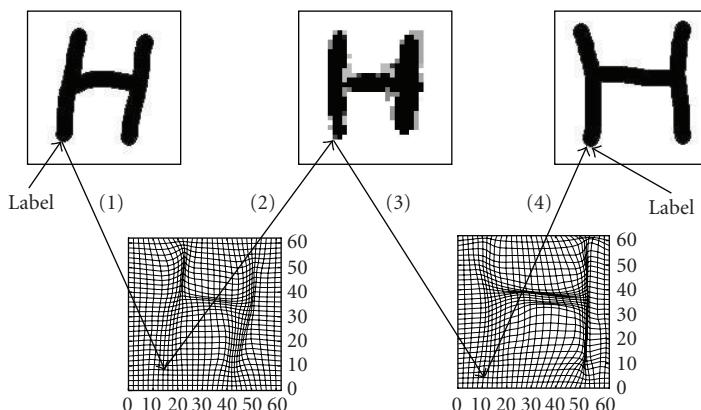


FIGURE 8.8. Results evaluation: a label marks a significant point of the image. At first (1) the match point which is closer to the label is identified. This allows to mark (2) an unit in the grid. A second image is presented to the network, and the unit identifies a new match point (3). The distance between the match point and a label (4) placed on the new image gives an estimate of the results.

automatically identifies distinctive features, performing a matching between the stored representation and the sample image.

However, the proposed architecture can still be enhanced in order to be adapted to different applications. For instance, an interesting field is the design of methods to optimize the elastic connections between units. A promising field seems to be the introduction of a multiscale representation in the feature maps, in order to improve the performances of the network.

Bibliography

- [1] A. A. Amini, T. E. Weymouth, and R. C. Jain, "Using dynamic programming for solving variational problems in vision," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 12, no. 9, pp. 855–867, 1990.
- [2] D. H. Ballard and C. M. Brown, *Computer Vision*, Prentice-Hall, Englewood Cliffs, NJ, USA, 1982.
- [3] L. Ballerini and L. Bocchi, "Multiple genetic snakes for bone segmentation," in *Applications of Evolutionary Computing (EvoWorkshop '03)*, vol. 2611 of *Lecture Notes in Computer Science*, pp. 346–367, Essex, UK, April 2003.
- [4] J. Buhmann, J. Lange, C. von der Malsburg, J. C. Vorbrüggen, and R. P. Würtz, "Object recognition with Gabor functions in the dynamic link architecture," in *Neural Network for Signal Processing*, pp. 121–156, Prentice-Hall, Englewood Cliffs, NJ, USA, 1992.
- [5] S. Cagnoni, A. B. Dobrzeniecki, R. Poli, and J. C. Yanch, "Genetic algorithm-based interactive segmentation of 3D medical images," *Image and Vision Computing*, vol. 17, no. 12, pp. 881–895, 1999.
- [6] T. F. Cootes, C. J. Taylor, D. H. Cooper, and J. Graham, "Image search using trained flexible shape models," in *Advances in Applied Statistics: Statistics and Images*, vol. 2, pp. 111–139, Carfax, Abingdon, UK, 1994.
- [7] J. D. Daugman, "Complete discrete 2-D Gabor transform by neural networks for image analysis and compression," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 36, no. 7, pp. 1169–1179, 1988.

- [8] R. C. Eberhart and Y. Shi, "Guest editorial: special issue on particle swarm optimization," *IEEE Transactions on Evolutionary Computation*, vol. 8, no. 3, pp. 201–203, 2004.
- [9] B.-C. Fan, C.-W. Hsieh, T.-L. Jong, and C.-M. Tiu, "Automatic bone age estimation based on carpal-bone image—a preliminary report," *Chinese Medical Journal*, vol. 64, no. 4, pp. 203–208, 2001.
- [10] D. Geiger, A. Gupta, L. A. Costa, and J. Vlontzos, "Dynamic programming for detecting, tracking, and matching deformable contours," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 17, no. 3, pp. 294–302, 1995.
- [11] U. Grenander and M. I. Miller, "Representations of knowledge in complex systems," *Journal of the Royal Statistical Society B*, vol. 63, no. 4, pp. 465–514, 1994.
- [12] W. W. Greulich and S. I. Pyle, *Radiographic Atlas of Skeletal Development of the Hand and Wrist*, Stanford University Press, Palo Alto, Calif, USA, 2nd edition, 1959.
- [13] R. P. Grzeszczuk and D. N. Levin, "Brownian strings: segmenting images with stochastically deformable contours," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 19, no. 10, pp. 1100–1114, 1997.
- [14] L. Ji and H. Yan, "Attractable snakes based on the greedy algorithm for contour extraction," *Pattern Recognition*, vol. 35, no. 4, pp. 791–806, 2002.
- [15] M. Kass, A. Witkin, and D. Terzopoulos, "Snakes: active contour models," *International Journal of Computer Vision*, vol. 1, no. 4, pp. 321–331, 1988.
- [16] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *Proceedings of IEEE International Conference on Neural Networks (ICNN '95)*, vol. 4, pp. 1942–1948, Perth, Australia, 1995.
- [17] T. Kohonen, *Self-Organization and Associative Memory*, Springer, New York, NY, USA, 3rd edition, 1989.
- [18] M. Lades, J. C. Vorbrüggen, J. Buhmann, et al., "Distortion invariant object recognition in the dynamic link architecture," *IEEE Transactions on Computers*, vol. 42, no. 3, pp. 300–311, 1993.
- [19] T. S. Lee, "Image representation using 2D Gabor wavelets," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 18, no. 10, pp. 959–971, 1996.
- [20] M. Niemeijer, B. van Ginneken, C. A. Maas, F. J. A. Beek, and M. A. Viergever, "Assessing the skeletal age from a hand radiograph: automating the tanner-whitehouse method," in *Medical Imaging: Image Processing*, vol. 5032 of *Proceedings of SPIE*, pp. 1197–1205, San Diego, Calif, USA, February 2003.
- [21] E. Pietka, A. Gertych, S. Pospiech, F. Cao, H. K. Huang, and V. Gilsanz, "Computer-assisted bone age assessment: image preprocessing and epiphyseal/metaphyseal ROI extraction," *IEEE Transactions on Medical Imaging*, vol. 20, no. 8, pp. 715–729, 2001.
- [22] G. Storvik, "A Bayesian approach to dynamic contours through stochastic sampling and simulated annealing," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 16, no. 10, pp. 976–986, 1994.
- [23] J. M. Tanner, R. H. Whitehouse, W. A. Marshall, and M. J. R. Healy, *Assessment of Skeletal Maturity and Prediction of Adult Height (TW2 Method)*, Academic Press, London, UK, 2nd edition, 1983.
- [24] C. von der Malsburg, "Pattern recognition by labeled graph matching," *Neural Networks*, vol. 1, no. 2, pp. 141–148, 1988.
- [25] D. J. Williams and M. Shah, "A fast algorithm for active contours and curvature estimation," *CVGIP: Image Understanding*, vol. 55, no. 1, pp. 14–26, 1992.
- [26] L. Wiskott, J.-M. Fellous, N. Krüger, and C. D. von Malsburg, "Face recognition by elastic bunch graph matching," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 19, no. 7, pp. 775–779, 1997.
- [27] R. P. Würtz, W. Konen, and K.-O. Behrmann, "On the performance of neuronal matching algorithms," *Neural Networks*, vol. 12, no. 1, pp. 127–134, 1999.

Leonardo Bocchi: Department of Electronics and Telecommunications, University of Florence,
Via di S. Marta 3, 50121 Florence, Italy

Email: leonardo.bocchi@unifi.it

9

Genetic snakes: active contour models by genetic algorithms

Lucia Ballerini

Genetic snakes are active contour models, also known as snakes, with an optimization procedure based on genetic algorithms. Genetic snakes are proposed to overcome some limits of the classical snakes, as initialization and existence of multiple minima, and successfully applied to segment different kinds of images. In this chapter, we review and extend the formulation of genetic snakes. New energy functionals are also described. Experimental results on synthetic images as well as on various real images are conducted with encouraging results.

9.1. Introduction

The active contour models or snakes [22] are an effective method to detect object boundaries in an image. The widely recognized power of deformable models is that they can exploit constraints derived from the image data along with a priori knowledge about the location, size, and shape of objects to segment. Originally developed for application to problems in computer vision and computer graphics, deformable models have been extensively applied in medical image analysis in problems including segmentation, shape representation, matching, and motion tracking, and have achieved considerable popularity.

However, the application of snakes to extract regions of interest suffers from some limitations. In fact, a snake is an energy minimizing spline and the classical model employs the variational calculus to iteratively minimize energy. There may be a number of problems associated with this approach such as algorithm initialization, existence of local minima, and selection of model parameters. Simulated annealing [18, 33], dynamic programming [2, 16], and greedy algorithms [21, 38] have been also proposed for minimization.

However, they are restricted both by the exhaustive searches for the admissible solutions and complicated parameter control and by the accurate initialization they require.

We propose the use of genetic algorithms (GAs) [17] to overcome some of the limits of the snake model. GAs offer a global search procedure that has shown its

robustness in many tasks. Their usefulness in pattern recognition and image processing has been demonstrated [1, 6]. The idea here is to address the initialization and the optimization of snake points through genetic minimization of the snake energy.

The purpose of this work is to review and extend the genetic snakes and to present additional energy functionals useful for specific applications. The organization of the chapter is as follows: in Section 9.2, we briefly review active contours, the basic notions, their limitations, and some improvements proposed in literature; in Section 9.3, we describe the genetic snake model; in Section 9.4, we discuss the new energy functionals and we present experimental results on synthetic images to test them. Several applications of genetic snakes to realistic problems are reported in Section 9.5, where we also illustrate new external image functionals specifically tailored for the applications under consideration.

9.2. Active contour models (snakes)

Due to the wide and successful application of deformable models, there exist survey papers focusing on different aspects of the model and its variants proposed in the literature [10, 20, 25, 28, 39].

Snakes are planar deformable contours that are useful in several image analysis tasks. They are often used to approximate location and shape of object boundaries on the basis of the reasonable assumption that boundaries are piecewise continuous or smooth.

Representing the position of a *snake* parametrically by $\mathbf{v}(s) = (x(s), y(s))$ with $s \in [0, 1]$, its energy can be written as

$$E_{\text{snake}} = \int_0^1 E_{\text{int}}[\mathbf{v}(s)]ds + \int_0^1 E_{\text{ext}}[\mathbf{v}(s)]ds, \quad (9.1)$$

where E_{int} represents the internal energy of the snake due to bending and it is associated with a priori constraints, E_{ext} is an external potential energy which depends on the image and accounts for a posteriori information. The final shape of the contour corresponds to the minimum of this energy.

In the original technique of Kass et al. [22], the internal energy is defined as

$$E_{\text{int}}[\mathbf{v}(s)] = \frac{1}{2} \left[\alpha(s) \left| \frac{\partial \mathbf{v}(s)}{\partial s} \right|^2 + \beta(s) \left| \frac{\partial^2 \mathbf{v}(s)}{\partial s^2} \right|^2 \right]. \quad (9.2)$$

This energy is composed of a first-order term controlled by $\alpha(s)$ and a second-order term controlled by $\beta(s)$. The two parameters $\alpha(s)$ and $\beta(s)$ dictate the simulated physical characteristics of the contour: $\alpha(s)$ controls the *tension* of the contour, while $\beta(s)$ controls its *rigidity*.

The external energy couples the snake to the image. It is defined as a scalar potential function whose local minima coincide with intensity extrema, edges, and other image features of interest. The external energy, which is commonly used to attract the snake towards edges, is defined as

$$E_{\text{ext}}[\mathbf{v}(s)] = -\gamma |\nabla G_\sigma * I(x, y)|^2, \quad (9.3)$$

where $G_\sigma * I(x, y)$ denotes the image convolved by a Gaussian filter with a standard deviation σ , ∇ is the gradient operator, and γ a weight associated with image energies.

The application of snakes and other similar deformable contour models to segment structures is, however, not without limitations. For example, snakes were designed as interactive models. In noninteractive applications, they must be initialized close to the structure of interest to guarantee good performance. The internal energy constraints of snakes can limit their geometric flexibility and prevent a snake from representing long tube-like shapes or shapes with significant protrusions or bifurcations. Furthermore, the topology of the structure of interest must be known in advance since classical deformable contour models are parametric and are incapable of topological transformations without additional model adjustments. Due to its own internal energy, the snake tends to shrink in case of lack of image forces, that is, constant image backgrounds or disconnected object boundaries.

Various methods have been proposed to improve and further automate the deformable contour segmentation process. A review of some of them can be found in [3] and in the above-mentioned surveys.

Few authors, to the best of our knowledge, propose the application of GAs to active contours. MacEachern and Manku [23] introduce the concept of active contour state and encode the variants of the state in the chromosome of the genetic algorithm. Tanatipanond and Covavisaruch [34] apply GAs to contour optimization with a multiscale approach. The fitness function is trained from a previously segmented contour, so that the system can learn the target's preselected features. Ooi and Liatsis [29] propose the use of coevolutionary genetic algorithms. They decompose the contour into subcontours and optimize each subcontour by separate GAs working in parallel and cooperating. Mishra et al. [27] apply GAs in contour extraction for noisy images. Echocardiographic images are preprocessed in order to find a rough boundary, GAs are then used to optimize this contour along radial search grid. However, in these approaches, the optimization is done in the neighborhood of the snake control points.

In other existing GA-based active contours, the optimization is done indirectly, that is, optimizing the parameters of the contour such as encoding the polygon [35], point distribution models [13, 31, 37], Fourier descriptors [14, 36], probability density functions [26] or edge detector, and elastic model parameters [8].

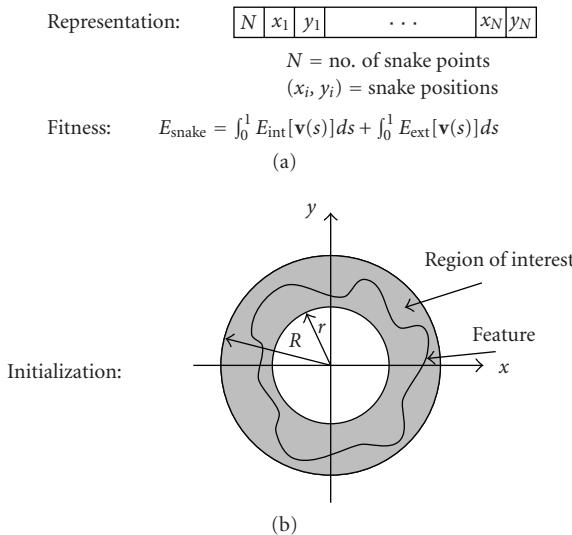


FIGURE 9.1. Genetic snakes: basic elements.

9.3. Genetic snakes

In this section, we review the genetic snake model, that is, our model of active contours, where the energy minimization procedure is based on GAs [4].

The basic elements of genetic snakes are solution representation, fitness definition, and initialization as illustrated in Figure 9.1. The parameters that undergo genetic optimization are the positions of the snake in the image plane $\mathbf{v}(s) = (x(s), y(s))$. The coordinates x and y are encoded in the chromosomes using a gray code. The total number of snake points is also encoded in the chromosomes and optimized using the GA. The user defines their range. The fitness function, that is to be minimized, is the total snake energy previously defined in (9.1), where E_{int} and E_{ext} are defined in (9.2) and (9.3).

The genetic optimization requires the definition of a region of interest given by r and R (the minimum and the maximum magnitude allowed for each $\mathbf{v}(s)$). The initial population is randomly chosen in such a region, and each solution lies in this region (r and R are user defined). This replaces the original initialization with a region-based version, enabling a robust solution to be found by searching the region for a global solution. Setting $r = 0$ and $R = \text{max}$ the solution is searched in the whole image, making initialization fully automatic.

The effect of the genetic operators on snake encoding is simple and intuitive (see Figure 9.2). Mutation randomly moves one snake point (x, y) to another point in the image plane. Crossover divides two parent snakes in two parts and recombines them to create two children snakes.

The GA implementation adopted in this work is GAucsd-1.4 [32]. Tests have shown that the model is not very sensitive to the control parameters of the GA. In our applications, we use most of the default parameters proposed by the GAucsd

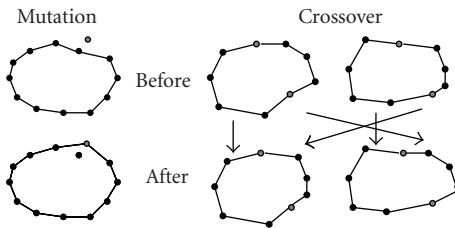


FIGURE 9.2. Genetic operators on snakes.

package, that is, gray code, fitness sigma scaling, two-point crossover, population size, and maximum number of generations computed according the length of the genome, roulette wheel selection. In case of use of different parameters, these are reported along with the experiment description. The main reason of using gray code is its simplicity. Further work using real encoding, which may provide a sub-pixellic positioning of the snake, can be done.

The genetic search strategy optimizes the snake model also in case of constant image background, where other optimization methods may fail, and overcomes difficulties related to spurious edge-points that can drive the snake to local minima. To reach an optimal minimum while avoiding local minima, alternative approaches suggest that the whole set of admissible curves be considered and the best one be chosen. Other methods are for local optimization, where only suboptimal solutions can be guaranteed. The GAs are particularly useful in simultaneously handling possible solutions and achieving a global minimum, while avoiding an exhaustive search.

It is known that the snake model requires either a local minimizer with good initialization or, otherwise, a global minimizer. Genetic snakes confront and overcome at the same time the two primary problems of initialization and optimization, and provide a global optimization with an automatic initialization.

9.4. Energy functionals

The classical optimization techniques impose different restrictions on the type of image functionals that can be employed (e.g., the existence of derivatives); the use of GAs gives us more freedom on the choice of such functionals. For this reason, we experiment new energy terms.

9.4.1. Internal energy

The internal energy term, E_{int} , controls the properties of the snake and it is expressed as defined in (9.2). This energy provides an efficient interpolation mechanism for recovering missing data. In our work, the values of the parameters $\alpha(s)$ and $\beta(s)$ are empirically chosen and they do not depend on the positions s of the snake, that is, we set $\alpha(s) = \alpha$ and $\beta(s) = \beta$, where α and β are constant values. In this way, different segments of the snake cannot have different elastic behavior.

The most adequate set of parameters for our snakes depends on several image characteristics. Therefore, given a particular application, some experimentation is required for choosing the best parameters. We performed experiments on synthetic images with different patterns and with additive noise having different variance.

The choice of the weights controls the type of solution produced by the active contour: large values of the weights associated with image functionals tend to move the snake boundary towards object contours, while large values of α and β increase smoothness and continuity. The signal-to-noise ratio (SNR) can affect the choice of weights: in low SNR images, or where missing and/or false edges are present, an increased contribution from the continuity and smoothness terms to the total energy is usually desirable. Large values for the continuity and curvature weights will discourage convergence to a “busy” contour, with notchings and indentations. On the other hand, small weights may allow the contour to be trapped into false edges or leak out through gaps in the boundary.

The first experiment uses synthetic images containing boundary of circles, squares, and star-shaped objects. The intensity images are generated by assigning grey level value 255 to pixels belonging to the boundary, and 100 otherwise. The boundary is smoothed by convolving the images with a 3×3 window who acts as a lowpass filter. These images are constructed to study the snake ability to capture corners as well as low curvature boundaries. Zero mean, white Gaussian noise is added to the images. Three different noise levels (corresponding to the standard deviation values: 20, 40, 60) are considered. This allows to study the robustness of our segmentation technique with respect to noise variance and to determine an adequate set of weights. Figure 9.3 shows some examples of simulated images.

On these images, we performed experiments using snakes having not more than 50 points, varying the energy weighting coefficients ($\alpha = 0.5, 0.8, 1, 1.2, 1.5, \beta = 0.5, 0.8, 1, 1.2, 1.5$, and $\gamma = 1$) running the GA for 23 00 000 iterations each time on a population of 10 000 individuals. The crossover rate and mutation rate are set respectively to 0.6 and 0.000006 based on experimental observations.

Figure 9.4 reports some of the results obtained. We observe that snakes with larger values of α and β have better noise rejection capabilities, but snakes having too large values of α and β tend to collapse on themselves. We also observe that snakes having a larger number of points are selected by our GA for the last set of images Figures 9.4(g), 9.4(h), and 9.4(i).

9.4.2. Area energy

The area energy is proportional to the area enclosed by the snake which has the effect of causing the contour to expand or contract. The area energy term we propose is

$$E_{\text{area}}[\mathbf{v}(s)] = \delta A, \quad (9.4)$$

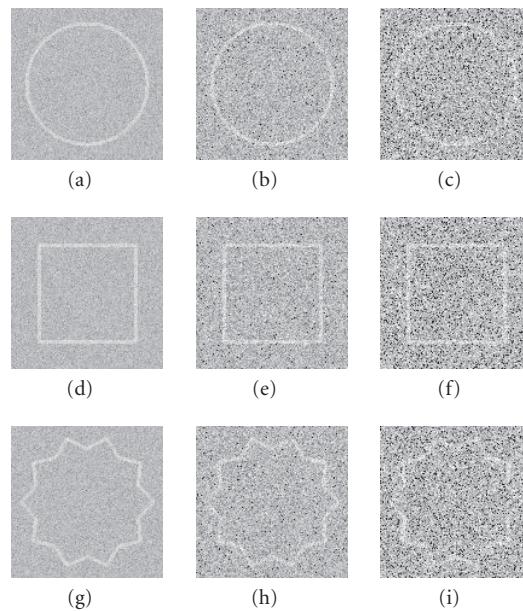


FIGURE 9.3. Examples of synthetic test images with different values of Gaussian noise (from left to right: $\sigma_{\text{noise}} = 20, 40, 60$).

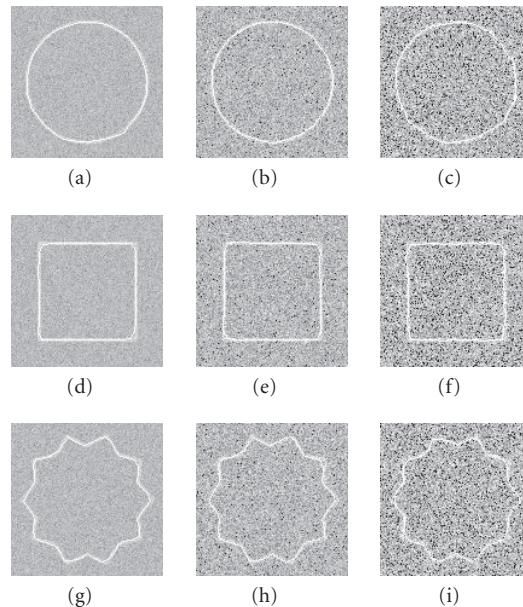


FIGURE 9.4. Simulation results on synthetic test images with different shapes, different values of Gaussian noise, and with the best set of snake coefficients.

where A is the area enclosed by the snake. The sign of δ determines whether the snake tends to expand or contract. For the case of a positive δ , this energy is a positive term in the fitness and causes the snake to choose regions enclosing small areas. On the other hand, for a negative δ , the snake tends to prefer regions having larger area.

The area is calculated as

$$A = \frac{1}{2} \left| \sum_{i=1}^N (x_i y_{i+1} - x_{i+1} y_i) \right|, \quad (9.5)$$

where (x_i, y_i) denotes the coordinates of a vertex and by convention $(x_{N+1}, y_{N+1}) = (x_1, y_1)$. This expression holds for any polygon provided that the vertices are ordered around the contour and no line segments joining the vertices intersect. Our representation of the coordinates, along with the use of polar coordinates, does not require any explicit check to ensure that this relationship is applicable.

The effect of this area energy is similar to the *balloon* force employed by L. D. Cohen and I. Cohen [11, 12]. In their model, they add an additional force in the normal direction of the snake. The implementation advantage is that in our formulation the area energy term depends on the snake positions, but not on derivatives.

We propose also an area energy which forces the snake to enclose a preset area:

$$E_{\text{area}}[\mathbf{v}(s)] = \delta (A - A_{\text{ref}})^2. \quad (9.6)$$

This energy has the form of a harmonic potential with a minimum when the area enclosed by the snake is equal to the reference area A_{ref} .

The experiment performed to show how the area energy works uses synthetic images containing circular shapes having an inner and an outer boundary as shown in Figure 9.5(a). The shape has grey level 100 on a black (grey level 0) background. These images are constructed to study the snake capability to be attracted by either the inner or the outer boundary according to the area energy term.

On these images, we performed experiments using snakes having 50 points, varying the energy weighting coefficients ($\alpha = 0.1, 0.15, 0.2, \beta = 0.1, 0.15, 0.2, \gamma = 0.1, 0.2, 0.3$, and $\delta = -0.3, -0.2, -0.1, 0.1, 0.2, 0.3$), running the GA for 2 000 000 iterations each time on a population of 10 000 individuals.

The results obtained with $\delta = +0.3$ and $\delta = -0.3$ are reported in Figures 9.5(b) and 9.5(c), respectively (other weight values are: $\alpha = 0.1, \beta = 0.2, \gamma = 0.2$). When the sign of δ is positive the snake chooses the inner boundary (b), while when it is negative the snake is attracted by the outer boundary (c).

9.4.3. Image energy

The external energy E_{ext} is composed by the image functionals. It is chosen according to the properties of the images of interest. The image functionals are designed to produce minima corresponding to target objects in the image. It is shown that

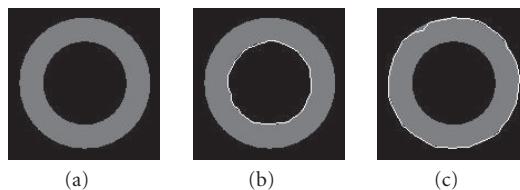


FIGURE 9.5. Simulation results: (a) example of synthetic test image, (b) genetic snake segmentation results with $\delta = +3$, and (c) $\delta = -3$.

the choice of the image functionals can affect the performance of the optimization technique. For this reason, we did experiment using various forms. For retinal images, we employ an image energy term which considers both the magnitude and the direction of the gradient and of the Laplacian of Gaussian. For color images, the image energy considers the gradient of the three RGB (red, green, blue) components separately.

As the external energy is strictly dependent on the specific application, we discuss image energy terms in the following section where we describe the applications of genetic snakes to realistic problems.

9.5. Applications

9.5.1. Medical images

Images used in this application are ocular fundus images, snakes are used in the segmentation of the foveal avascular zone (FAZ). Diabetic retinopathy is the leading cause of new adult blindness; one way to early detect diabetic retinopathy is the study of the FAZ. In fact, retinal capillary occlusion produces an FAZ enlargement. Moreover, the FAZ is characterized by qualitative changes showing an irregular contour with notches and indentations [7]. The segmentation of FAZ boundary is usually considered the starting point for this analysis.

Retinal images are taken by a Scanning Laser Ophthalmoscope, with a frequency of 25 frames per second following the injection of a bolus of fluorescein. These images are digitized into 512×512 pixel matrices with 256 gray levels per pixel. The region of interest, that is, the FAZ, is approximately in the center of these images. For simplicity, the origin of the coordinates, that is, the snake center, is located at the center of the FAZ. Its position can be chosen approximately by the user. The image energy functionals are chosen according to FAZ properties as follows.

First, we consider a functional which localizes bright lines since FAZ boundaries are ultimately bright lines (i.e., capillaries) with an intensity maximum at their center. A simple external energy functional that attracts a snake towards lines is the image intensity:

$$E_{\text{img}}[\mathbf{v}(s)] = \gamma I(x, y), \quad (9.7)$$

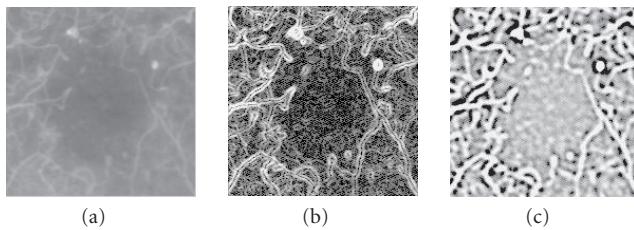


FIGURE 9.6. Retinal images: (a) image intensity, (b) image convolved by gradient of Gaussian ($\sigma = 2$), (c) image convolved by Laplacian of Gaussian ($\sigma = 2$).

where γ is a weight factor whose sign determines whether the snake is attracted by dark or bright lines. For the case of a negative γ , the snake is attracted to local minima of E_{img} , which corresponds to local maxima of intensity, that is, bright lines. This functional (see Figure 9.6(a)) can detect roof edges. For our purpose this functional localizes the medial axis of the capillaries. However, the achievable performances are only partially satisfactory, due to the adjacency of the snake to the bigger vessels exhibiting a strong maximum; moreover, dye leakage introduces a light haze with consequent artifacts on the image function.

Then, we consider a functional which attracts the snake towards image edges, that is, in our case, vessel boundaries. In this case, if edges are of interest, the image energy is defined as

$$E_{\text{grad}}[\mathbf{v}(s)] = -|\nabla I(x, y)|^2, \quad (9.8)$$

where $\nabla I(x, y)$ is the gradient of the image. An easy implementation of this functional can be obtained by computing the gradient of Gaussian (GoG) of the image intensity:

$$E_{\text{GoG}}[\mathbf{v}(s)] = -|\nabla G_\sigma * I(x, y)|^2. \quad (9.9)$$

The resulting image functional is shown in Figure 9.6(b). The weight in this case is negative so that local minima of E_{GoG} correspond to maxima of the gradient, that is, strong edges. Simple use of this functional for FAZ boundary extraction also does not give fully satisfactory performance. The fact that it is the edge of the vessels that is localized and not the point of maximum intensity provides a basis for uncertainty.

This suggests that we consider both the gradient magnitude and the gradient direction of the image. A suitable functional may be obtained by constructing the dot product of the contour tangent with the normalized gradient vector:

$$E_{\text{dir}}[\mathbf{v}(s)] = \left| \frac{\partial \mathbf{v}}{\partial s} \cdot \frac{\nabla I(x, y)}{|\nabla I(x, y)|} \right|. \quad (9.10)$$

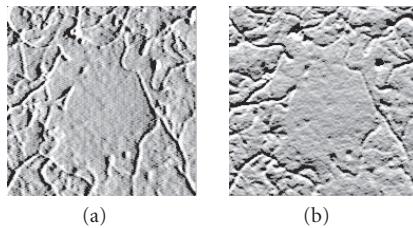


FIGURE 9.7. The x and y components of the gradient of the image. The intensity of each pixel is proportional to the gradient component in that point.

The weight of this factor is positive, so that orientation inconsistencies tend to be penalized. Anyway, edge points whose orientation disagrees with that of the overlaying snake may also yield minimal values of the external energy. Hence, the snake is able to discriminate against phantom lines, while allowing for the presence of corners. The two components of the gradient are shown in Figure 9.7.

In order to increase the locus of attraction of a minimum, we experiment with a slightly different edge functional (also proposed by Kass et al. [22]):

$$E_{\text{LoG}}[\mathbf{v}(s)] = -|\nabla^2 G_\sigma * I(x, y)|^2. \quad (9.11)$$

Minima of this functional lie on zero-crossings of $\nabla^2 G_\sigma * I(x, y)$ which defines edges in the Marr-Hildreth theory [24]. This image functional is shown in Figure 9.6(c).

In addition, since image gradient and Laplacian of Gaussian (LoG) produce random edges in the background region where some noise is present, we can improve FAZ boundary localization by including a Gaussianly smoothed version of the image intensity (with large σ).

Thus, the proposed image energy is composed of four terms and is expressed as

$$\begin{aligned} E_{\text{ext}}[\mathbf{v}(s)] = & -\gamma_1 G_\sigma * I(x, y) - \gamma_2 |\nabla G_\sigma * I(x, y)|^2 \\ & + \gamma_3 \left(\mathbf{n} \cdot \frac{\partial \mathbf{v}}{\partial s} \right) - \gamma_4 |\nabla^2 G_\sigma * I(x, y)|^2, \end{aligned} \quad (9.12)$$

where $\mathbf{n} = \nabla G_\sigma * I(x, y) / |\nabla G_\sigma * I(x, y)|$.

In this application, the internal energy weights α and β are normally kept constant while the image energy weights are varied to find a good balance between the four terms. We observe that values of α and β close to 1 give good results. The values of the weights associated with image functionals are chosen in the range [0.5, 0.8].

In Figure 9.8, we can see some examples of retinal images with FAZ outlines segmented by our snake model.

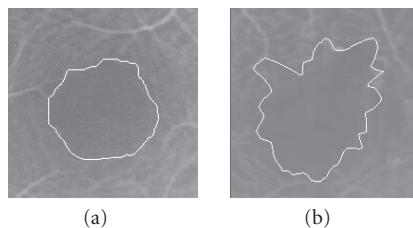


FIGURE 9.8. Images of (a) normal and (b) diabetic FAZ with superimposed genetic snakes.

9.5.2. Radar images

The aim of this application is to determine the shape and size of aircrafts in advanced surface movement guidance and control systems (A-SMGCS). Genetic snakes are used to segment and detect target moving along runways and taxiways of an airport from images provided by a surface movement radar, even with a very noisy image.

The images used in this application are radar images obtained by courtesy of Oerlikon Contraves Italiana SpA. The radar sensor is a prototype operating in the millimeter band (95 GHz) [15]. The images are recorded in the range/azimuth reference system, directly from the video output of the radar receiver. From these images, smaller areas are extracted around positions where most probably an aircraft is present [30]. The target is extracted by resampling the radar image in a rectangular grid of known resolution.

It is important to notice that the target echoes do not resemble exactly the real shape of the target itself. The appearance of the target in a radar image depends on its electromagnetic properties, on the radar resolution, and on the position of the target with respect to the sensor. Usually the target appears on the image as a fragmented shape, reflecting a multipoint scattering model, in which the echo is supposed to be due to a number of points, scattered all over the target itself. Radar images taken into account have poor dynamic range, so target and background clutter have often a similar echo strength. For instance, in several images the aircraft wings are undistinguishable from the grass below. An adaptive filtering scheme, based upon real-time histogram analysis, is applied to the images in order to enhance the contrast and to let the next steps of the algorithm work in better conditions.

Several image processing algorithms can be used to get some parameters about the airplane from the image. Among the others, parameters such as approximate dimensions, shape, or directions are very useful in order to guess what it is and what it is doing. We apply genetic snakes to segment and detect these aircraft parameters.

The energy functionals used in this application include the gradient of Gaussian and the Laplacian of Gaussian of the enhanced image.

In Figure 9.9, we can see an example of original image, an edge detector results, and the corresponding aircraft outlines segmented by our genetic snake model.

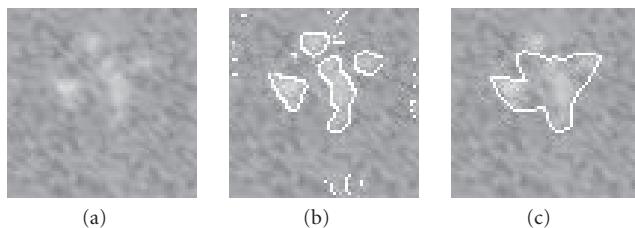


FIGURE 9.9. (a) Original image containing the echo from a B747 and strong clutter. (b) The output of an edge extractor algorithm. (c) The image segmented with our method.

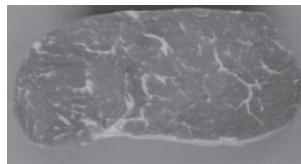


FIGURE 9.10. Digital camera photograph of a representative beef meat (in original color).

In this application, we can observe that the advantage of genetic snakes over methods like edge extractor operators is their flexibility to operate on fragmented shapes which are not as easy with convolutional operators.

9.5.3. Food images

Genetic snakes are here applied to meat images in order to segment them, with the special purpose of separating connective tissue from the remaining parts of the meat.

In another work, we have developed a method to solve the specific problem of measuring the percentage of fat [5]. In particular, a color segmentation algorithm to classify different substances as muscle, fat, and connective tissue has been optimized for camera photographs of meat. The classification is fully automatic and combines a fuzzy clustering algorithm, the fuzzy c -means algorithm with a GA. There is still an open problem with these images: fat and connective tissue present almost the same color and are therefore almost indistinguishable by any color segmentation technique, therefore the idea of using genetic snakes.

Color images of many samples of beef meat are captured by a Sony DCS-D700 camera. Images are 1024×1024 pixel matrices with a resolution of 0.13×0.13 mm. They are represented in an RGB (red, green, and blue) format (see Figure 9.10).

Images are preprocessed to suppress background; the choice of green color as background was very helpful for this stage. Three thresholds (RGB) are used in this phase. An example of background suppression is shown in Figure 9.11. Then we apply genetic snakes to segment meat from connective tissue.

In order to apply genetic snakes to color images, we employ a modified version of the image energy. When the goal is to fit a snake to a boundary within an image,

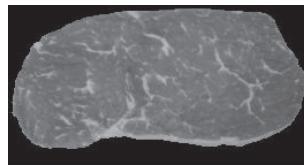


FIGURE 9.11. Background suppression from image shown in Figure 9.10.

it is useful to preprocess the image with an edge detector so that the points of maximum gradient are emphasized. One of the most commonly employed edge detectors uses the gradient of the image convolved with a Gaussian smoothing function.

Here we consider the gradient of the three-color RGB components, that yields an image energy functional composed of three terms:

$$\begin{aligned} E_{\text{img}}[\mathbf{v}(s)] = & -\gamma_R |\nabla G_\sigma * I_R(x, y)|^2 \\ & - \gamma_G |\nabla G_\sigma * I_G(x, y)|^2 \\ & - \gamma_B |\nabla G_\sigma * I_B(x, y)|^2, \end{aligned} \quad (9.13)$$

where $I_R(x, y)$, $I_G(x, y)$, $I_B(x, y)$ are the three components of image intensities. The weights are negative so that local minima of E_{img} correspond to maxima of the gradient, that is, strong edges.

We observe that the connective tissue is usually located close to the border of meat and that there is a strong edge between it and the muscle. Moreover, the percentage of connective tissue is usually around 4 ÷ 5%. Thus, the proposed area energy functional is expressed as

$$E_{\text{area}}[\mathbf{v}(s)] = \delta(A - 0.95A_{\text{ref}})^2, \quad (9.14)$$

where A_{ref} is the reference area computed on the background suppressed images.

Hence, we use a snake energy composed of three terms: the internal energy, E_{int} , defined in (9.2), the image energy, E_{img} , defined in (9.13), and the area energy, E_{area} , defined in (9.14). We use snakes having 50 points, running the GA for 40 000 000 iterations on a population of 100000 individuals. Crossover and mutation rate are set, respectively, to 0.43 and 0.000001. Weight values are $\alpha = 0.15$, $\beta = 0.15$, $\gamma_R = \gamma_G = \gamma_B = 0.3$, and $\delta = 0.2$.

In Figure 9.12, we can see the segmentation results obtained by our genetic snake model. Note that the connective tissue in the bottom part of the image is separated from the rest of the meat.

9.6. Conclusions

In this chapter, we have discussed genetic snakes and presented an extension of our previous model. The present method deals with two important and difficult problems of the conventional snakes, namely, initialization and optimization.

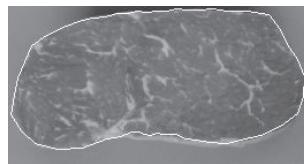


FIGURE 9.12. Digital camera photograph of a representative beef meat with superimposed the final position of our genetic snake model.

The energy minimization procedure based on GAs overcomes the problems associated with sensitivity to initialization and local minima, which are crucial problems of classical techniques. Genetic snakes do not need to be initialized close to the structure of interest to guarantee good performance. Indeed, all possible initial choices can be considered within the GA framework, where the user has only to define the region where the search takes place. Local minima, caused by spurious edges or constant image background, are easily avoided by the genetic optimization. Furthermore, the new model is better at extracting nonconvex shapes compared to conventional snakes.

We have also described additional energy terms, like an area energy and a modified version of the image energy, which accounts for both the magnitude and the direction of the gradient and the Laplacian of Gaussian, that exhibits interesting properties in the localization of FAZ boundary. Other energy terms may be easily added using the genetic optimization procedure. This is not possible in the classical snake formulation that requires energy to be a differentiable function. In our model, there is no restriction on the form of the energy functionals.

The real-world experiments show that the new model works well for contour-based segmentation in a variety of image analysis applications. The results reported are the best results obtained in several runs (usually 10 for each set of experiment and value of the parameters). Statistical analysis of them proved their consistency and repeatability. The main problem with this method is the time required and the impossibility to exploit the ability of GAs to generalize, as each individual image needs to be processed from scratch.

In this work, we applied GAs to the positions of the snake. The management of the weights for different energy terms is still an open important problem and it is usually fulfilled empirically, except for a promising emerging idea suggested in [9]. Their framework consists in a learning section and a detection section, and provides a training mechanism to obtain the weights from a desired object contour given manually. This mechanism first employs Taguchi's method to determine the weight ratios among distinct energy terms, followed by a weight refinement step with a GA. Further work on this technique could be the study of the evolution of the parameters and the functionals governing the snake behavior; and possibly discovery of new snake-like models as, for example, the deformable organisms proposed by [19].

At this stage, we used genetic snakes for segmenting a single object in the image. The case of multiple objects is a challenging problem. Future studies could

consider parallel optimization of multiple intercommunicating genetic snakes, as well as an extension to 3D.

Acknowledgments

The first genetic snake model and the application to retinal images are part of the author's Ph.D. thesis, carried out at the Department of Electronic Engineering, University of Florence, Italy. The application to radar images is a collaboration with Enrico Piazza, presently at Thales ATM, Gorgonzola, Milan, Italy. The application to meat images is performed at the Centre for Image Analysis, Uppsala, Sweden, in collaboration with the Department of Food Science, SLU, Uppsala, Sweden.

Bibliography

- [1] J. T. Alander, "Indexed bibliography of genetic algorithms in signal and image processing," Tech. Rep. 94-1-SIGNAL, Department of Information Technology and Production Economics, University of Vaasa, Vaasa, Finland, 1995.
- [2] A. A. Amini, T. E. Weymouth, and R. C. Jain, "Using dynamic programming for solving variational problems in vision," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 12, no. 9, pp. 855–867, 1990.
- [3] L. Ballerini, *Computer aided diagnosis in ocular fundus images*, Ph.D. thesis, Università di Firenze, Firenze, Italy, 1998.
- [4] L. Ballerini, "Genetic snakes for medical images segmentation," in *Proceedings of the 1st European Workshops on Evolutionary Image Analysis, Signal Processing and Telecommunications (EvoWorkshops '99)*, vol. 1596 of *Lectures Notes in Computer Science*, pp. 59–73, Göteborg, Sweden, May 1999.
- [5] L. Ballerini, A. Höglberg, K. Lundström, and G. Borgefors, "Color image analysis technique for measuring of fat in meat: an application for the meat industry," in *Machine Vision Applications in Industrial Inspection IX*, vol. 4301 of *Proceedings of SPIE*, pp. 113–124, San Jose, Calif, USA, January 2001.
- [6] C. Bounsaythip and J. Alander, "Genetic algorithms in image processing—a review," in *Proceedings of the 3rd Nordic Workshop on Genetic Algorithms (3NWGA '97)*, pp. 173–192, Helsinki, Finland, August 1997.
- [7] G. H. Bresnick, R. Condit, S. Syrjala, M. Palta, A. Groo, and K. Korth, "Abnormalities of the foveal avascular zone in diabetic retinopathy," *Archives of Ophthalmology*, vol. 102, no. 9, pp. 1286–1293, 1984.
- [8] S. Cagnoni, A. B. Dobrzeniecki, R. Poli, and J. C. Yanch, "Genetic algorithm-based interactive segmentation of 3D medical images," *Image and Vision Computing*, vol. 17, no. 12, pp. 881–895, 1999.
- [9] D.-H. Chen and Y.-N. Sun, "A self-learning segmentation framework—the Taguchi approach," *Computerized Medical Imaging and Graphics*, vol. 24, no. 5, pp. 283–296, 2000.
- [10] K.-W. Cheung, D.-Y. Yeung, and R. T. Chin, "On deformable models for visual pattern recognition," *Pattern Recognition*, vol. 35, no. 7, pp. 1507–1526, 2002.
- [11] L. D. Cohen, "On active contour models and balloons," *CVGIP: Image Understanding*, vol. 53, no. 2, pp. 211–218, 1991.
- [12] L. D. Cohen and I. Cohen, "Finite-element methods for active contour models and balloons for 2-D and 3-D images," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 15, no. 11, pp. 1131–1147, 1993.
- [13] T. F. Cootes, C. J. Taylor, D. H. Cooper, and J. Graham, "Active shape models—their training and application," *Computer Vision and Image Understanding*, vol. 61, no. 1, pp. 38–59, 1995.

- [14] Y. Fan, T. Jiang, and D. J. Evans, "Volumetric segmentation of brain images using parallel genetic algorithms," *IEEE Transactions on Medical Imaging*, vol. 21, no. 8, pp. 904–909, 2002.
- [15] M. Ferri, G. Galati, F. Marti, P. F. Pellegrini, and E. Piazza, "Design and field evaluation of a millimetre-wave surface movement radar," in *Proceedings of the IEE Radar Conference (RADAR '97)*, pp. 6–10, Edinburgh, UK, October 1997.
- [16] D. Geiger, A. Gupta, L. A. Costa, and J. Vlontzos, "Dynamic programming for detecting, tracking, and matching deformable contours," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 17, no. 3, pp. 294–302, 1995.
- [17] D. E. Goldberg, *Genetic Algorithms in Search, Optimization, and Machine Learning*, Addison-Wesley, Reading, Mass, USA, 1989.
- [18] R. P. Grzeszczuk and D. N. Levin, "Brownian strings: segmenting images with stochastically deformable contours," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 19, no. 10, pp. 1100–1114, 1997.
- [19] G. Hamarneh, T. McInerney, and D. Terzopoulos, "Deformable organisms for automatic medical image analysis," in *Proceedings of the 4th International Conference on Medical Image Computing and Computer-Assisted Intervention (MICCAI '01)*, pp. 66–75, Utrecht, The Netherlands, October 2001.
- [20] A. K. Jain, Y. Zhong, and M.-P. Dubuisson-Jolly, "Deformable template models: a review," *Signal Processing*, vol. 71, no. 2, pp. 109–129, 1998.
- [21] L. Ji and H. Yan, "Attractable snakes based on the greedy algorithm for contour extraction," *Pattern Recognition*, vol. 35, no. 4, pp. 791–806, 2002.
- [22] M. Kass, A. Witkin, and D. Terzopoulos, "Snakes: active contour models," *International Journal of Computer Vision*, vol. 1, no. 4, pp. 321–331, 1988.
- [23] L. A. MacEachern and T. Manku, "Genetic algorithms for active contour optimization," in *Proceedings of IEEE International Symposium on Circuits and Systems*, vol. 4, pp. 229–232, Monterey, Calif, USA, 1998.
- [24] D. Marr, *Vision*, W. H. Freeman, New York, NY, USA, 1980.
- [25] T. McInerney and D. Terzopoulos, "Deformable models in medical image analysis: a survey," *Medical Image Analysis*, vol. 1, no. 2, pp. 91–108, 1996.
- [26] M. Mignotte, C. Collet, P. Pérez, and P. Bouthemy, "Hybrid genetic optimization and statistical model-based approach for the classification of shadow shapes in sonar imagery," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 22, no. 2, pp. 129–141, 2000.
- [27] A. Mishra, P. K. Dutta, and M. K. Ghosh, "A GA based approach for boundary detection of left ventricle with echocardiographic image sequences," *Image and Vision Computing*, vol. 21, no. 11, pp. 967–976, 2003.
- [28] J. Montagnat, H. Delingette, N. Scaple, and N. Ayache, "Representation, shape, topology and evolution of deformable surfaces. Application to 3D medical image segmentation," Tech. Rep. RR-3954, INRIA, Sophia Antipolis, France, 2000.
- [29] C. Ooi and P. Liatsis, "Co-evolutionary-based active contour models in tracking of moving obstacles," in *Proceedings of International Conference on Advanced Driver Assistance Systems*, pp. 58–62, Birmingham, UK, September 2001.
- [30] E. Piazza, "Adaptive algorithm for real-time target extraction from a surface movement radar," in *Parallel and Distributed Methods for Image Processing IV*, vol. 4118 of *Proceedings of SPIE*, pp. 58–66, Denver, Colo, USA, July 1999.
- [31] C. F. Ruff, S. W. Hughes, and D. J. Hawkes, "Volume estimation from sparse planar images using deformable models," *Image and Vision Computing*, vol. 17, no. 8, pp. 559–565, 1999.
- [32] N. N. Schraudolph and J. J. Grefenstette, "A user's guide to GAucsd 1.4," Tech. Rep. CS92-249, Computer Science and Engineering Department, University of California, San Diego, La Jolla, Calif, USA, 1992.
- [33] G. Storvik, "A Bayesian approach to dynamic contours through stochastic sampling and simulated annealing," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 16, no. 10, pp. 976–986, 1994.

- [34] T. Tanatipanond and N. Covavisaruch, "An improvement of multiscale approach to deformable contour for brain MR images by genetic algorithms," in *Proceedings of IEEE International Symposium on Intelligent Signal Processing and Communication Systems (ISPACS '99)*, pp. 677–680, Phucket, Thailand, December 1999.
- [35] A. Toet and W. P. Hajema, "Genetic contour matching," *Pattern Recognition Letters*, vol. 16, no. 8, pp. 849–856, 1995.
- [36] P. E. Undrill, K. Delibasis, and G. G. Cameron, "An application of genetic algorithms to geometric model-guided interpretation of brain anatomy," *Pattern Recognition*, vol. 30, no. 2, pp. 217–227, 1997.
- [37] K.-W. Wan, K.-M. Lam, and K.-C. Ng, "An accurate active shape model for facial feature extraction," *Pattern Recognition Letters*, vol. 26, no. 15, pp. 2409–2423, 2005.
- [38] D. J. Williams and M. Shah, "A fast algorithms for active contours and curvature estimation," *CVGIP: Image Understanding*, vol. 55, no. 1, pp. 14–26, 1992.
- [39] C. Xu, D. L. Pham, and J. L. Prince, "Image segmentation using deformable models," in *Handbook of Medical Imaging*, M. Sonka and J. Fitzpatrick, Eds., vol. 2, chapter 3, pp. 129–174, SPIE Press, Bellingham, Wash, USA, 2000.

Lucia Ballerini: European Centre for Soft Computing, Mieres, 33600 Asturias, Spain

Email: lucia.ballerini@softcomputing.es

10

Visual texture classification and segmentation by genetic programming

Vic Ciesielski, Andy Song, and Brian Lam

10.1. Introduction

While there is a considerable history of work on visual texture, the definition of texture is still imprecise. However, it is generally agreed that a texture is spatially homogeneous and contains repeated visual patterns. In synthetic textures, such as horizontal lines, vertical lines, or a checkerboard, the basic structure is repeated exactly. In natural textures, such as grass, wood, sand, or rocks, there is some random variation in size, shape, intensity, or colour in the repetitions of the basic structure. Figure 10.1 shows some examples of artificial (a,b) and natural (c,d) textures.

Texture information is potentially very useful in computer vision applications such as image/video retrieval, automated industrial inspection, and robot navigation. However, currently deployed systems do not use texture, primarily because current algorithms result in unacceptably long computation times. Fast and accurate texture recognition could have a major impact on the design of future vision systems.

From the perspective of computer vision there are two main problems relating to texture: classification and segmentation. The classification task assumes we have images of single textures and we are required to distinguish them. For example, if we have images like Figures 10.1(c) and 10.1(d), or subimages cut out from them, we need to identify which are grass and which are sand. In a texture segmentation task, we have images containing several textures and are required to identify the regions in the image occupied by each texture, as shown in Figure 10.2. The input image contains arbitrary regions of two different textures and the segmented image is a two-color image with one colour used for each texture. A problem in which we are required to retrieve all images containing, for example, a sandy beach are a variant of the segmentation problem in which only one texture is of interest.

Texture classification and segmentation problems can be supervised, where the set of textures is known in advance, or unsupervised, where it is not known. This paper is concerned with supervised situations.

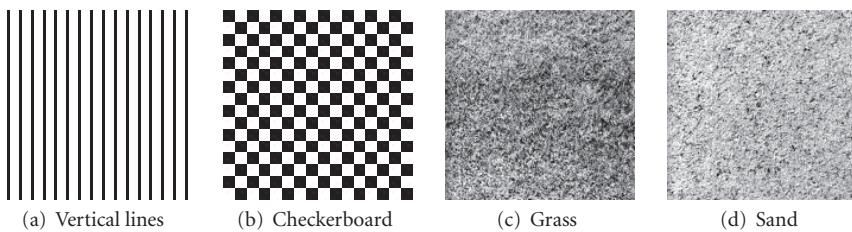


FIGURE 10.1. Examples of synthetic and natural textures.

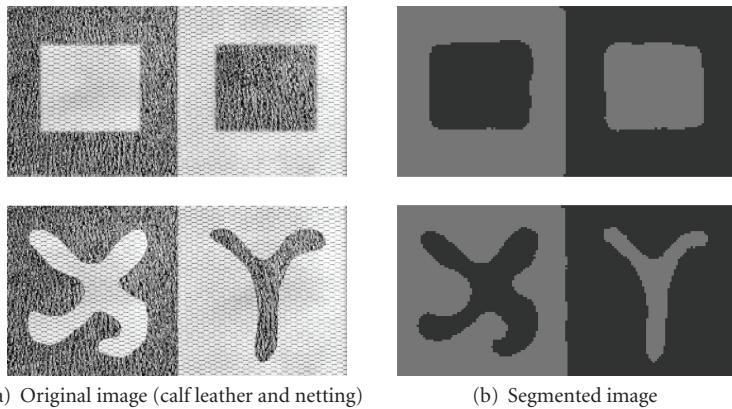


FIGURE 10.2. Examples of segmentation by texture.

10.1.1. Conventional approach to texture classification

The conventional approach to texture classification is shown in Figure 10.3. The task is to assign a texture label to an image like one of those in Figure 10.1. A two-step procedure is used. In the first step, a vector of features, based on human derived theories and models of texture, is computed. In the second step, a classifier such as a decision tree, neural network, or nearest neighbour classifier is used to assign a class to the feature vector (a large number of classifiers and their implementations are described in [30]). Most of the research in texture classification is focused on the first stage and a large number of different ways of getting useful, highly discriminatory features have been investigated. These include Haralick features [10], Laws masks [16], and various wavelet transforms [4, 9, 17, 26]. The most widely used features over the last 35 years are the Haralick features which are based on an intermediate data structure called the grey level cooccurrence matrix. Over the years there have been many refinements to the basic Haralick approach, for example, [19]. More recently, features based on wavelets have generated considerable interest. Most new algorithms for texture features are compared against the Haralick features. Work on texture up until 1993 is reviewed in [28], the most

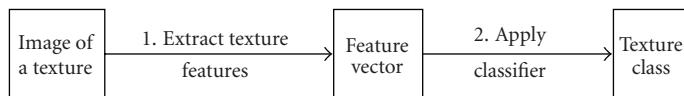


FIGURE 10.3. The conventional approach to texture classification.

recent review of texture analysis. In some very recent work, there has been a focus on developing models of a texture unit or “texton” and using the models to develop classifiers [8, 31]. However, expensive computation is required as various Gabor and Laplacian transforms at a number of locations, scales, and orientations are used.

The conventional approach has three main drawbacks. Firstly, there is no universal set of optimal texture features. Some of the features work very well for some textures and very badly for others. While there are many texture features, it is not clear which combination of features will suit a given problem and a trial and error process is needed for each new texture classification/segmentation task. Secondly, some of the approaches generate an enormous number of features, perhaps more than there are pixels in the image. This necessitates complex dimensionality reduction in feature space. Thirdly, most of the texture feature extraction algorithms are computationally expensive. They require the generation of Fourier-type transforms or other complex intermediate data structures and then additional computation on these structures. In this paper, we show that the use of the genetic programming techniques can overcome some of these drawbacks.

10.1.2. Aim

The aim of this paper is to describe a number of ways in which genetic programming can be used in texture classification and to show how some of the classifiers can be used for fast, accurate texture segmentation.

There are at least three ways in which genetic programming can be used in texture classification. They are as follows.

- (1) Use conventional feature extraction followed by evolving a genetic programming classifier, that is, use a genetic programme as a classifier in step 2 of Figure 10.3.
- (2) Use a one-step procedure in which the classifiers are evolved directly from example images of the textures of interest, that is, replace steps 1 and 2 with a single step which does the classification directly from the pixels without any feature extraction.
- (3) Evolve the actual feature extraction programmes which can then be used with a conventional classifier or a GP classifier, that is, replace the human constructed feature extraction programmes in step one with evolved feature extraction programmes.

In Sections 10.3 to 10.5, we describe these approaches. In Section 10.6, we show how the fastest classifiers can be used for segmentation.

All of the texture images used in this paper are taken from the Brodatz album [5]. The original photographs were taken by a commercial photographer, Phil Brodatz, and intended for creative designers. The photographs have been digitized and have become a kind of de facto standard in texture research. We have used the images from [1]. Altogether there are 111 different textures labeled D1 to D112, with D14 missing.

10.2. Genetic programming

Genetic programming is a methodology for obtaining computer programmes to solve a particular problem by a process of simulated evolution. An initial population of programmes is constructed. Each programme is executed on the problem at hand and its success on the task, its fitness, is measured. A new population of programmes is then constructed by selecting the fitter programmes as parents and generating children by recombining selected parts of the parents (crossover) and/or making random changes to the parents (mutation). This process continues until the problem is solved or until some preset number of generations has been completed. If the process is working well, the programmes will gradually become fitter and fitter through the generations until the problem is solved.

At the current state of the art the evolved programmes are not in conventional programming languages such as C or Java. Rather, they are in languages which have been designed with restricted syntax and semantics so that two arbitrarily combined programme fragments will form a syntactically valid executable programme. Approaches include tree-based genetic programming in which the programmes are in a subset of LISP, linear genetic programming in which the programmes are in an assembly language, and Cartesian genetic programming in which the evolved programme is a network of specialised computing nodes. Our work uses tree-based genetic programming.

In tree-based genetic programming, programmes are represented as tree structures. An example tree and the corresponding code are shown in Figure 10.4. The internal nodes are functions and the terminals are inputs to the programme. The tree is evaluated in a bottom up fashion and the value of the root node is the output of the programme.

In specifying the configuration of a genetic programming run it is necessary to give the functions, the terminals, a method of evaluating fitness, and a number of parameters for the evolutionary parameters. These include the population size, the maximum number of generations to compute if a solution is not found, the elitism rate (the percentage of best individuals in the current generation copied without change to the next generation), the crossover rate (percentage of individuals in the new population that are created by crossover), mutation rate (the percentage of individuals in the new population created by mutation), and the maximum permitted tree depth.

While there has been previous work on texture recognition using evolutionary computation techniques, for example, [27], genetic programming for image

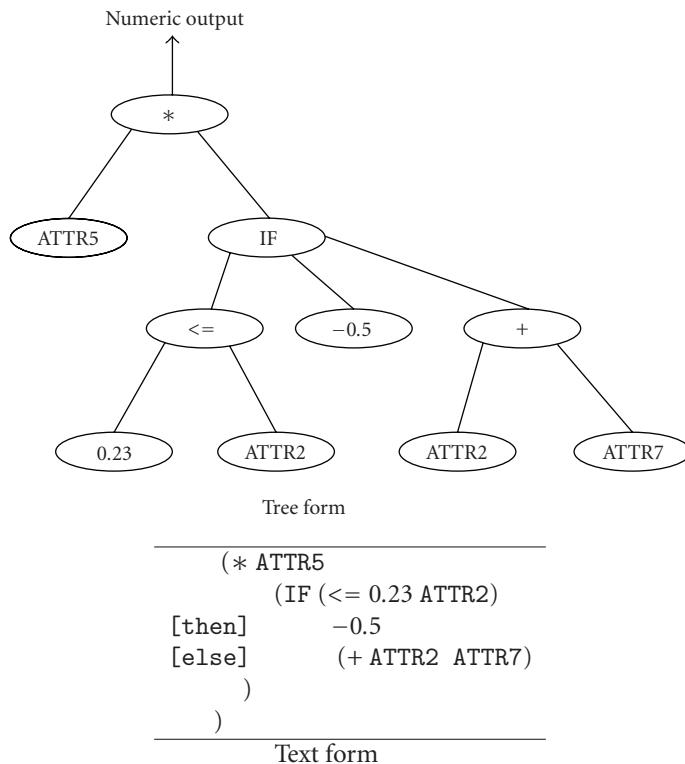


FIGURE 10.4. A programme in tree-based genetic programming.

processing tasks [21, 22], and genetic programming for texture synthesis [12], there is no significant prior work on genetic programming for texture recognition.

10.2.1. Genetic programming classifiers

There has been prior work on evolving genetic programming classifiers [14, 18]. For a problem in which there are only two classes, for example, images of grass and sand, the most straightforward approach is to use positive values of a programme like the one shown in Figure 10.4 as texture 1, for example, grass, and negative outputs as texture 2, for example, sand. To obtain the classifier the available data is split into a training and test set, following the machine learning methodology for learning from examples. To get the fitness of an evolved programme, it is applied to each example in the training data and the number of classification errors is counted. The fewer the errors, the fitter the programme. Once a programme achieves a fitness of zero the evolutionary run can be terminated.

We have used a refinement of this basic method which is described in [18, 25]. In this refinement, called dynamic range selection, the real line is split into a variable number of ranges, not just two (less than 0, greater than 0) as in the straightforward approach. The range boundaries are evolved along with the classification

```

(* -0.208533 (if (n= (- ATTR39 ATTR69) (+ ATTR114 0.299995)) (- ATTR222
0.253174) (+ (+ (if (Between (* ATTR189 (+ ATTR205 ATTR74)) ATTR164
ATTR12) ATTR108 (if (Between (- (* 0.253174 ATTR129) 0.9875) 0.128228
(/ ATTR189 0.940897)) (+ ATTR39 (if (Between -0.947054 ATTR164 ATTR12)
ATTR160 ATTR124)) (* 0.5979 0.230855))) (+ (+ (if (Between (+ ATTR205
ATTR74) ATTR164 ATTR12) ATTR108 (if (Between -0.947054 (if (Between
-0.947054 ATTR164 ATTR12) ATTR160 ATTR124) ATTR12) (+ ATTR39 (if (
Between -0.947054 ATTR164 (- (/ ATTR149 (if (Between -0.947054 (if (
Between -0.947054 ATTR164 ATTR12) ATTR160 ATTR124) ATTR12) ATTR108
0.135714)) (* ATTR164 ATTR129))) ATTR160 ATTR124)) (* 0.5979 0.230855)))
(if (Between (* ATTR189 (+ (if (Between (- (* 0.253174 ATTR129) 0.9875)
0.128228 (/ ATTR189 (- 0.987674 0.747389))) (+ ATTR39 (if (Between
-0.947054 ATTR164 ATTR12) ATTR160 ATTR124)) (* 0.5979 0.230855)))
ATTR74)) ATTR164 ATTR12) ATTR205 (+ (+ (if (Between (* (* ATTR189 (+
ATTR205 ATTR74)) (+ ATTR205 ATTR74)) ATTR164 ATTR12) ATTR108 (if
Between (- (* 0.253174 ATTR129) 0.9875) 0.128228 (/ ATTR189 (- 0.987674
0.747389))) (+ ATTR39 (if (Between -0.947054 ATTR164 ATTR12) ATTR160
ATTR124)) (* 0.5979 0.230855))) (+ (+ ATTR160 (if (Between -0.947054
ATTR164 ATTR12) ATTR160 ATTR114)) (* 0.253174 ATTR129)) ATTR205)) (*
0.253174 ATTR129)) ATTR205)))

```

Ranges:

Class 1: D112 : -131 ~ -107 and 24 ~ 136 and 175 ~ +250

Class 2: Other Textures: -250 ~ -132 and -106 ~ 23 and 137 ~ 184

FIGURE 10.5. A typical evolved programme. ATTR_x is the value of the pixel at position $((x-1) \bmod 16, (x-1)/16)$ of the input image.

programme. In Figure 10.5, for example, if the output of the programme is less than -250, then the example is class 2; if the output of the programme is between -131 and -107, it is class 1. These classifiers are more accurate and are evolved in fewer generations than the ones from the straight forward approach [18]. Also, they can be easily extended to more than two classes.

10.3. Two-step texture classification: GP classifier

In this approach conventional texture feature extraction programmes are used to generate a feature vector, but a genetic programming classifier will be used in second step of the process described in Figure 10.3.

In determining which features to use there are thousands of options. As indicated in Section 10.1.1 there have been many papers on approaches to extracting visual texture features, some of which generate hundreds of features. In our experiments, we have used a selection of Haralick [10] and Gabor wavelet features [9, 17].

We used thirteen Haralick feature functions. They are (1) angular second moment, (2) contrast, (3) correlation, (4) variance, (5) inverse difference moment, (6) sum average, (7) sum variance, (8) sum entropy, (9) entropy, (10) difference variance, (11) difference entropy, and (12) and (13) mean of correlation. Every

TABLE 10.1. Functions used in the genetic programming runs.

Name	Return type	Argument types	Description
+	Dbl	Dbl Dbl	Arithmetic addition
-	Dbl	Dbl Dbl	Arithmetic subtraction
×	Dbl	Dbl Dbl	Arithmetic multiplication
%	Dbl	Dbl Dbl	Protected division
IF	Bool	Bool Dbl Dbl	If arg1 is true return arg2 else return arg3
\leq	Bool	Dbl Dbl	True if $\text{arg 1} \leq \text{arg 2}$
\geq	Bool	Dbl Dbl	True if $\text{arg 1} \geq \text{arg 2}$
=	Bool	Dbl Dbl	True if $\text{arg 1} = \text{arg 2}$
Between	Bool	Dbl Dbl Dbl	True if the value of arg1 is between arg2 and arg3

function is computed at four angles (0° , 45° , 90° , and 135°) for five distance values (1, 3, 5, 7, 9). At each distance, the average of each feature for the four angles is used as an additional feature. This gives a total of $13 \times 4 \times 5 = 260$ Haralick features.

For the Gabor features, we have used five fragment sizes from 2×2 to 32×32 and six orientations (0° , 30° , 60° , 90° , 120° , 150°). For each combination of scale and orientation, the mean and the standard deviation of the transform coefficients are computed. Thus there are $5 \times 6 \times 2 = 60$ Gabor features extracted from each image.

In total, the feature vector for each image contains $260 + 60 = 320$ elements, which form the terminal set. The function set is shown in Table 10.1.

10.3.1. Configuration of genetic programming

We have performed 110 experiments on the Brodatz textures to determine how well one texture can be distinguished from the other 110. The details of the experiments are given below. The images used have been randomly cut out from the original 640×640 Brodatz texture images from [1]. The experiments have been done using the RMIT-GP package [2].

Images: 1110 subimages from the chosen texture make up class 1 and 10 random subimages from each of the other 110 make up class-2, giving a total of 2210 examples per experiment.

Image size: 64×64 pixels.

Number of classes: 2.

Error estimation: 10-fold cross-validation.

Functions as shown in Table 10.3.

Terminals: as shown in Table 10.2, the values of each extracted feature, together with a random number generator, comprise the terminals.

Fitness: classification error.

TABLE 10.2. Terminal set.

Name	Return type	Description
Random($-1, 1$)	Double	Random constant
Feature[x]	Double	Value of feature x , $0 \leq x \leq 295$

TABLE 10.3. Comparison of classification accuracy on textures pictured in this chapter.

No.	Texture	FE + C4.5		FE + GP		One step GP	
		Train	Test	Train	Test	Train	Test
D2	Fieldstone	98.72	92.14	93.79	93.15	84.26	80.38
D3	Reptile skin	99.77	98.03	97.65	96.80	95.56	94.13
D4	Pressed cork	99.86	98.85	99.67	99.54	86.55	86.10
D5	Expanded mica	99.22	94.01	93.79	91.78	83.28	82.76
D9	Grass	99.26	94.74	93.40	89.49	82.04	81.44
D12	Bark	99.49	93.74	91.90	89.95	82.76	74.72
D15	Straw	99.81	97.89	98.82	98.63	90.53	90.93
D16	Herringbone	99.81	98.76	99.41	99.54	84.53	80.81
D19	Woollen Cloth	99.26	94.74	93.01	92.69	84.72	84.78
D21	French canvas	99.95	99.77	100.00	100.00	97.58	96.12
D24	Calf Leather	99.77	98.12	98.23	98.17	89.62	89.46
D29	Beach Sand	99.72	96.71	97.06	95.89	88.31	84.34
D34	Netting	100.00	99.90	100.00	100.00	99.28	99.55
D38	Water	99.72	97.57	98.36	97.26	96.60	95.84
D57	Straw matting	99.86	99.26	99.73	99.08	91.25	87.13
D68	Wood grain	99.72	98.81	99.73	99.08	95.69	94.44
D84	Raffia	99.81	97.53	98.75	98.63	93.79	94.42
D92	Pigskin	99.54	96.75	95.88	95.89	83.81	82.36
D94	Brick Wall	99.45	96.84	97.51	96.80	93.14	92.19
D112	Plastic Bubbles	99.49	95.38	96.54	92.23	88.44	87.17
	Average of 110		95.67		94.17		87.86

Parameters: population size, 200; max generations, 50; elitism rate, 0.10; crossover rate, 0.85; mutation rate, 0.05; maximum tree depth, 30.

10.3.2. Results

The results for the textures pictured in this chapter are given in Table 10.3. The reasons for choosing these textures are given in Section 10.5. For each classifier 10 runs were performed and the performance of the best evolved classifier is shown in the two columns labeled FE + GP (feature extraction, genetic programming classifier). The averages over all 110 experiments are shown as the last line. For comparison, the results of using the same features with a well known, commonly used classifier, the C4.5 decision tree classifier, are shown in the columns labeled FE + C4.5. The J48 implementation of the C4.5 algorithm in the Weka machine

learning toolkit was used with the default parameters [3]. The complete results for all 111 textures can be found in [23].

Over the entire image set the C4.5 classifier is slightly more accurate. However, there was considerably more over training with C4.5 than with the genetic programming classifier. The major drawback of the GP approach is the training time, which is considerably longer than C4.5.

Interestingly, when the same experiment was repeated with a set of synthetic binary textures such as Figures 10.1(a)-10.1(b), the GP classifier achieved 100% accurate classification on all test data while the C4.5 classifier made a small number of errors.

10.4. One-step texture classification

In this section, we look at a novel approach to performing texture classification in one step. The texture images are given directly to the GP system and the classifiers are evolved from the image pixels directly, bypassing the feature extraction step.

10.4.1. Configuration of genetic programming

Number of images: as in Section 10.3.1.

Image size: 64×64 as in Section 10.3.1 except that the images were scaled to 1/4 of their original size.

Error estimation: 10-fold cross-validation.

Functions: as shown in Table 10.3.

Terminals: as shown in Table 10.2, except that there are 256 terminals and the terminal “Attribute[x]” returns the intensity value of the pixel at position $((x - 1) \bmod 16, (x - 1)/16)$ of the input image.

Fitness: classification error.

Parameters: population size, 200; max generations, 50; elitism rate, 0.10; crossover rate, 0.85; mutation rate, 0.05; maximum tree depth, 30.

Scaling of the images to 16×16 results in 256 terminals in a genetic programming run. Unscaled images would require $64 \times 64 = 4096$ terminals. This is beyond the capacity of our genetic programming system. We have found empirically that around 500 terminals is the limit. This is due to the increase in the size of the search space because of the large number of possibilities whenever it is necessary to choose a terminal.

The classification results for this approach for a sample of the 111 textures are shown in the last column of Table 10.3. In 18% of the experiments, or 20 cases, accuracies above 95% were achieved. Of these 20 cases, five were classified nearly perfectly with accuracy above 99%. None of the cases reached 100% test accuracy. In three cases early termination occurred, that is, the evolutionary process terminated with 100% classification accuracy on the training data before 50 generations was reached. The one step classifiers tended to be more accurate when the texture images are roughly homogeneous in terms of the structure of the texture or intensity distribution and when geometric regularity (e.g., bricks) was present. The one

step classifiers were least accurate when the images are extremely inhomogeneous and the subimages cut from different parts of the original image vary significantly.

It can be seen from Table 10.3 that the one-step classifiers are generally less accurate than classifiers based on extracted features. This is not unexpected as the feature extraction algorithms are based on well-thought out human models and theories. What is surprising is that the evolved classifiers which are based solely on image pixels are so accurate over such a large variety of natural textures. In addition, the evolved classifiers are very fast compared to those using feature extraction. They involve the evaluation of a relatively simple arithmetic expression based on the values of a relatively small number of pixels. This suggests that these classifiers could be useful for texture segmentation in algorithms that require a classifier to be executed at each pixel position. The decreased accuracy could perhaps be offset by the knowledge that neighbouring pixels are likely to be in the same texture region. We come back to this issue in Section 10.6.

A typical-evolved programme is shown in Figure 10.5. The programmes varied in size, but no successful programme contained more than 700 nodes. A negative aspect is that the programmes are too complex comprehend and the underlying algorithm cannot be recovered. It is not possible to tell whether some real texture regularities have been captured or whether some artifact of the training data has been captured. In order to determine whether any texture regularities are being captured in the evolved programmes we have carried out a number of experiments with simple binary textures involving horizontal, vertical, and diagonal lines. We have used a small number of terminals and functions and encouraged the evolution of small programmes by using a size penalty in the fitness function. Analysis of the evolved programmes revealed that the evolved programmes could be interpreted as texture masks which did capture regular differences between the textures. This work is described in [23, 24]. This result, together with the large number of training images used with the Brodatz textures, gives us confidence that texture regularities are being discovered even though the programmes are too complex for analysis.

10.5. Two-step texture classification: evolved features

In this section, we return to the two-step procedure. However, in this case we replace the hand-crafted texture feature extraction programmes by evolved programmes. The feature extraction programmes will be evolved from a selection of textures which we call the learning set. The evolved features will then be evaluated using a train-and-test methodology on the learning set and on a problem involving a different set of Brodatz textures.

Following the successful use of direct pixel inputs in the one-step approach, our original intention was to use pixel inputs from the examples of the learning set for the evolution of the feature extraction programmes. However, feature extraction programmes evolved in this fashion were not very discriminatory. From an analysis of the evolved programmes and results it appeared that some kind of aggregation of pixel information was needed. We subsequently experimented with

histograms, which give a very coarse aggregation, and discovered that this worked surprisingly well. The methodology and results are described in this section.

The histogram of an 8-bit grey level image is a vector of length 256 in which element i gives the number of pixels which have a grey level value of i . In a histogram all information about the spatial arrangement of pixels is lost, which suggests that histograms might not be very useful for texture.

To evolve the feature extraction programmes we have selected a learning set of 13 textures, shown in Figures 10.11(a)–10.11(m). These textures were chosen to enable comparison with [29] in which 18 texture feature extraction algorithms are compared on a set of classification problems using this data set.

We have evolved 78 feature extraction programmes, one from each pair of textures shown in Figure 10.11. The evolutionary procedure described in the next section was repeated 78 times, once for each pair of combinations of the 13 textures in the learning set.

10.5.1. Configuration of genetic programming

Images: two textures selected from the 13 shown in Figure 10.11.

Image size: 64×64 cut randomly from the original large images.

Number of images: 80 images of texture 1 and 80 of texture 2.

Functions: one only $\{+\}$. We began with the function set used in the earlier experiments (Table 10.3). However, we found that using fewer operators resulted in programmes that were just as accurate, but which were easier to comprehend.

Terminals: histogram $[i]$, $0 \leq i \leq 255$.

Fitness evaluation: a feature extraction algorithm is considered useful if the feature values result in high classification accuracy. This will occur if the feature values computed for each class are well separated in feature space. Thus, to evolve feature extraction algorithms, we need a way to implement the intuition that “the better the separation, the better the fitness.” We have done this by computing the overlap between clusters generated by the K -means clustering algorithm. An example of this, for the case where there are two texture classes in the learning set, is shown in Figure 10.6. To get the data shown in the figure, a programme in the population has been evaluated on a learning set of 160 images which consist of 80 examples of texture (a) from Figure 10.11 and 80 examples from texture (b). The averages of the feature values for each class give cluster centroids at 561 and 323. The mid point of the two centroids is the cluster boundary, that is, 443. There are 4 cluster1 feature values below the boundary and 6 cluster2 features above it, thus 10 points are incorrectly clustered. Equivalently, it can be considered that there are 10 errors. This represents quite good discrimination. In contrast, if there were 150 errors, the discrimination would be very poor.

Parameters: population size, 100; max generations, 200; elitism rate, 0.02; crossover rate, 0.78; mutation rate, 0.28; maximum tree depth, 9.

10.5.2. Results

The 78 feature extraction programmes were used on a 4-class problem involving the textures shown in Figure 10.7 to generate a feature vector of length 78. Note

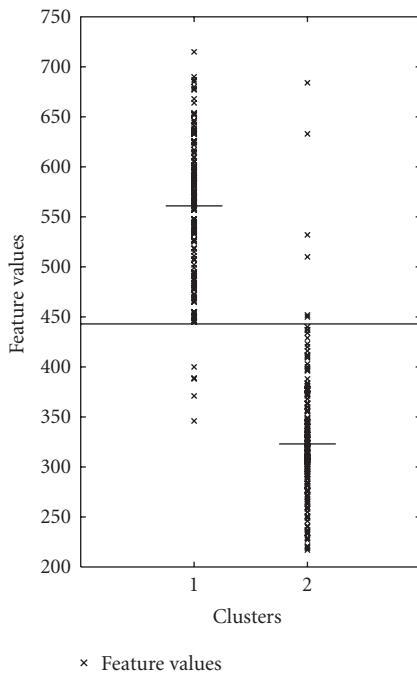


FIGURE 10.6. Feature space for two texture classes, D9 and D12.

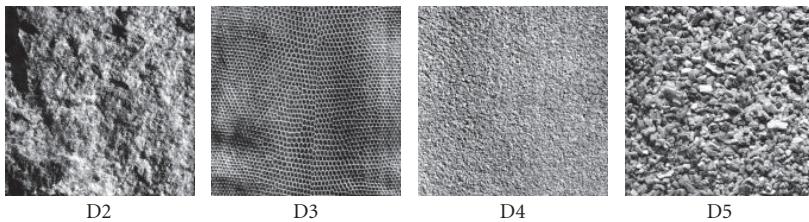


FIGURE 10.7. Four Brodatz textures used for testing evolved features.

that these are not in the learning set. There were 33 training images and 67 test images. Using a nearest neighbour classifier, the evolved features gave a test accuracy of 100% while the Haralick features gave a test accuracy of 95.5%. This result suggests that some generalization has occurred, the evolved programmes have captured some texture regularities that apply not just to the learning set but to other textures as well.

We have tested our evolved features on the two benchmark texture classification problems used in [29]. In [29], 18 human derived texture feature extraction algorithms are compared on two problems. The first problem is a 13-class problem involving the textures shown in Figures 10.11(a)–10.11(m). On this problem the evolved features outperformed 5 of the human derived methods. The second

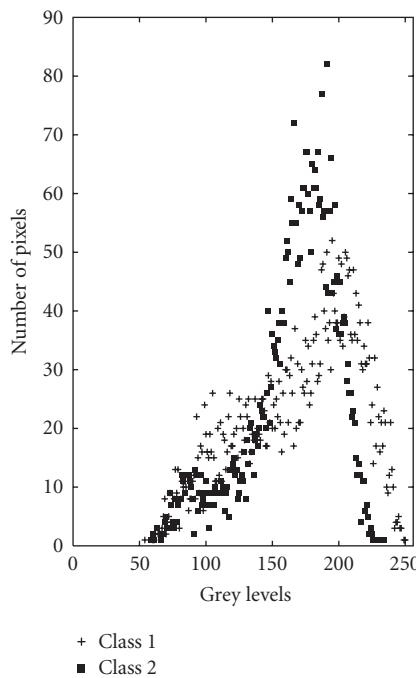


FIGURE 10.8. Histograms of D9 (Class1) and D12 (Class2) textures.

problem is a 15-class problem involving textures from a different texture data base, the Vistex data base. Vistex problems are generally considered to be more difficult than Brodatz problems. This is because textures in a class are at different resolutions. In the Brodatz textures, images with three different sizes of bricks would be three different texture classes. In the Vistex textures they are one class. On this problem the evolved features outperformed 8 of the human derived methods. Full details of the results are in [15].

Using only histograms means that spatial relationships between pixels are being ignored. Two quite different textures could give similar histograms. Ways of incorporating spatial relationships still need to be investigated.

10.5.3. Analysis of evolved feature extraction programmes

Since $+$ is the only function, all of the evolved algorithms are sums of the number of pixels at certain grey levels. For example, using grass (D9) as class1 and bark (D12) as class2 gives the feature extraction programme $X109 + 2 * X116 + 2 * X117 + X126 + 2 * X132 + X133 + 2 * 143 + X151 + X206 + X238 + 3 * X242 + X254$, where $Xnnn$ represents the value of the histogram at grey level nnn . If we examine the histograms of two images, shown in Figure 10.8, we can see the programme has made use of the points between grey level 100 and grey level 150 and those above grey level 200 where class1 is significantly different from class2.

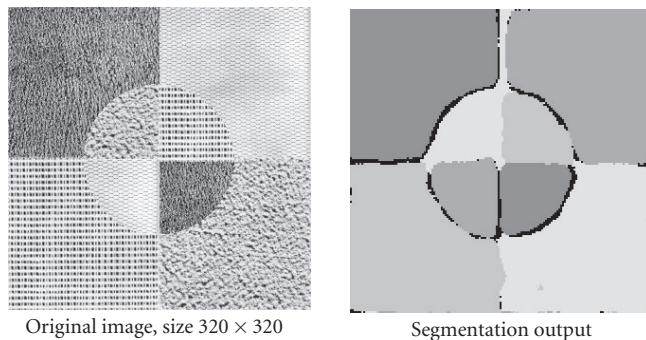


FIGURE 10.9. Segmentation of a mosaic of four textures, D21, D24, D34, D57.

10.6. Texture segmentation

The one step texture classifiers described in Section 10.4 can be readily adapted for texture segmentation. If it is known that there are only two textures in the image to be segmented, as in Figure 10.2 for example, then a binary classifier can be evolved from small windows cut out from each of the textures. In the segmentation step the classifier is then evaluated at each pixel position in the image to be segmented and each pixel given a colour based on the output of the classifier. This is what has been done to achieve the segmentation shown in Figure 10.2. A window size of 16 was used.

The choice of window size is important, it cannot be too small or too big. Since a texture T is composed of repeated visual patterns, subimage should also exhibit the texture. If we take smaller and smaller cutouts, these should still be texture T . However, the cutout cannot be too small, if the cutout is smaller than the basic repetitive unit, we no longer have texture T . For example, if we cut Figure 10.1(d) into quarters, then we still have a recognisable sand texture. If we cut those quarters into quarters, we still have sand. However, if we cut too many times, a subimage will no longer look like sand. As another example, consider bricks. As long as the cutout is at least as big as a brick we still have a brick texture; if it is smaller, the regular rectangular pattern of a brick texture is lost. In evolving one step classifiers for a segmentation task it is important that the window size is not smaller than the size of the basic repeating unit. If the window size is too big, there will be many ambiguous pixel labels around the boundaries between textures. For natural textures such as those in the Brodatz album it is difficult to determine an optimal window size a priori. We have empirically fixed on 16.

If there are more than two textures in the image to be segmented, as in the left-hand image of Figure 10.9 where there are four, the segmentation algorithm is more complex. We need 4 “me-versus-everybody-else” classifiers of the kind described in Sections 10.3 and 10.4. Each of the classifiers is evaluated at each pixel position, with the pixel of interest at the centre of the window. Each classifier labels each pixel as “my-texture” or “not-my-texture.” The final label or colour is

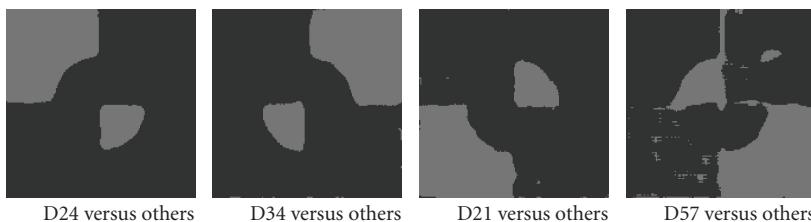


FIGURE 10.10. Areas claimed by each classifier: (light: “my-texture,” dark: “not-my-texture”).

then determined as follows: if a pixel has 4 “not-my-texture” labels, it is rendered in black. If it has 3 “not-my-picture” labels and 1 “my-texture” label, then it is labeled and rendered as the claiming texture. If it is claimed by 2 or more textures, it receives the label of the classifier that achieved the highest accuracy on the training data. The output of this algorithm is shown in the right-hand image of Figure 10.9. In Figure 10.10 the areas claimed by each of the classifiers are shown in grey and the areas not claimed are shown in black. It can be seen that all but the last are quite accurate. Even though the last classifier has made some mistakes, the final segmentation is still very accurate because of the voting procedure.

10.6.1. Segmentation speed

The classifiers generated by the single-step approach are relatively small. No classifier for Brodatz textures was observed to have more than 700 functions, which are simple operators like $+$, $-$, $*$, $/$, and IF. This characteristic enables a fast voting strategy which can lead to much faster segmentation than the conventional “feature extraction plus classification” approaches. In contrast, feature extraction algorithms are much more complex. Using Haralick features [11] as an example, at least one 256×256 cooccurrence matrix needs to be generated in computing the features for an image with 256 gray levels. One of the simplest Haralick features, energy, is computed as

$$\sum_{i=0}^{255} \sum_{j=0}^{255} M^2(i, j), \quad (10.1)$$

where $M(i, j)$ denotes one cell of the matrix. Such a calculation requires 256×256 repetitions of $+$ and $*$. Therefore, more than 100 000 operations are needed. Thus extracting Haralick features is much more expensive than executing GP generated classifiers. Even if the construction of the cooccurrence matrix and the cost of the subsequent classification step of the conventional approach required zero computation, the single-step classifier would still be much faster than the two-step approach.

In a texture segmentation task the feature extraction and classification must be performed many times. Consider a situation where the binary segmentation algorithm described above is being used. We have measured the time for performing

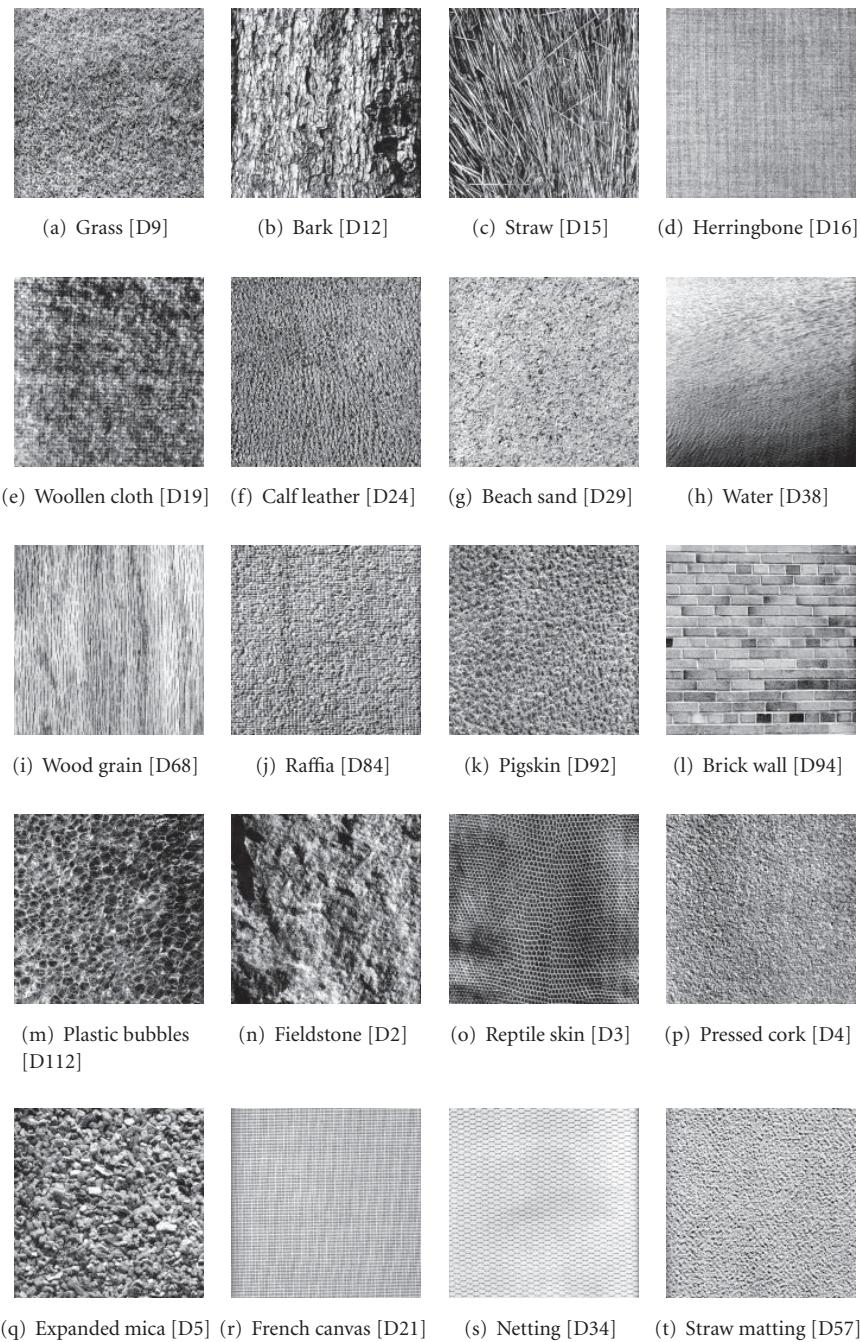


FIGURE 10.11. The Brodatz textures used in this paper.

one calculation of the Haralick features on a Sun SPARC computer and found that it is approximately 0.7 seconds. This means that the time to perform a segmentation of 256×256 gray level image using a window size of 16×16 is approximately

$$256^2 \times 0.7 \approx 45\,000 \text{ s.} \quad (10.2)$$

There are various ways of speeding up the approach, some heuristics are described in [20]. One method is to evaluate only every 2nd or 3rd pixel position. Another is to use a split-and-merge approach where the image is recursively split into quarters to some predetermined size and the texture label of each fragment computed. Adjacent areas of the same texture are then merged into one region. If the split is limited to only 3 levels, features are required to be computed on four 128×128 quarters for the first split, on sixteen 64×64 smaller quarters for the second split and on sixty-four 32×32 squares for the third split. The total CPU time for computing these features can be estimated as

$$4 \times 4.58 + 16 \times 4.10 + 64 \times 1.70 = 192.72 \text{ s.} \quad (10.3)$$

The times for computing Haralick features for images of size 128×128 , 64×64 , and 32×32 on the SPARC computer are 4.58, 4.10, and 1.70 seconds, respectively.

Clearly, whether the voting strategy or the split-and-merge strategy is used, the times taken on the SPARC computer for computing these features are much longer than 2.58 seconds, the time for segmenting a 320×320 image by our proposed algorithm. Furthermore, it is not hard to see that with only three levels of split, where 32×32 is the smallest size, the output will be too coarse to produce smooth boundaries. Further splitting requires even more computation time. To achieve an equivalent segmentation performance, the time spent by a conventional method based on Haralick features or Gabor features would be more than one hundred times the time spent by our method. There are many reports in the literature which describe long computation times for texture segmentation. Jain and Karu reported 122 seconds of processing time on a SUN SPARC-10 workstation to segment a 256×256 gray-level image using Gabor filters and 109 seconds processing time by their proposed method [13]. Chang et al. evaluated different texture segmentation algorithms and reported the run times of “feature extraction plus classification” on a Sun Ultra SPARC. All of the reported times were more than 28 minutes [6]. In 2002, Chen and Chen proposed a new feature for fast texture segmentation [7]. The runtime for their experiment of segmenting 256×256 images on a Pentium III-500 PC was about 1 minute. Although the run times reported in the literature are measured under different conditions and not suitable for direct comparison, it is clear that the proposed method requires much less computation time since there is no computational expensive feature extraction.

10.7. Conclusions

We have described 3 different ways in which genetic programming can be used in texture classification: (1) as the classifier in the conventional “feature extraction +

classification" approach, (2) as a one-step classifier which bypasses feature extraction and uses pixel values directly, and (3) as a method for evolving texture feature extraction programmes to be used in the classical two-step procedure. Overall the genetic classifiers are competitive with conventional classifiers in terms of accuracy. The major drawback is that they require long computation times to get a classifier. However, once evolved, the classifiers are very fast. This result suggests that the genetic classifiers could be used in real-time situations where speed is more important than accuracy. A drawback of the genetic programming classifiers is that the programmes are hard to comprehend.

An exciting outcome of this work is the accuracy and speed of the one-step classifiers. This offers the prospect of fast real-time segmentation in application areas such as robot vision, industrial inspection, and image/video retrieval, applications in which texture is currently not being used because the computational cost is too high.

Bibliography

- [1] Brodatz texture database, <http://www.ux.his.no/~tranden/brodatz.html>.
- [2] RMIT-GP: The RMIT University Genetic Programming System, School of Computer Science and Information Technology. <http://www.cs.rmit.edu.au/~vc/rmitgp>.
- [3] Weka machine learning project, Department of Computer Science, The University of Waikato, <http://www.cs.waikato.ac.nz/~ml/>.
- [4] S. Arivazhagan and L. Ganesan, "Texture classification using wavelet transform," *Pattern Recognition Letters*, vol. 24, no. 9-10, pp. 1513-1521, 2003.
- [5] P. Brodatz, *Textures: A Photographic Album for Artists and Designers*, Dover, New York, NY, USA, 1966.
- [6] K. I. Chang, K. W. Bowyer, and M. Sivagurunath, "Evaluation of texture segmentation algorithms," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 1, pp. 294-299, Ft. Collins, Colo, USA, June 1999.
- [7] K.-M. Chen and S.-Y. Chen, "Color texture segmentation using feature distributions," *Pattern Recognition Letters*, vol. 23, no. 7, pp. 755-771, 2002.
- [8] O. G. Cula and K. J. Dana, "3D texture recognition using bidirectional feature histograms," *International Journal of Computer Vision*, vol. 59, no. 1, pp. 33-60, 2004.
- [9] S. E. Grigorescu, N. Petkov, and P. Kruizinga, "Comparison of texture features based on Gabor filters," *IEEE Transactions on Image Processing*, vol. 11, no. 10, pp. 1160-1167, 2002.
- [10] R. M. Haralick, "Statistical and structural approaches to texture," *Proceedings of the IEEE*, vol. 67, no. 5, pp. 786-804, 1979.
- [11] R. M. Haralick, K. Shanmugam, and I. Dinstein, "Textural features for image classification," *IEEE Transactions on Systems, Man and Cybernetics*, vol. 3, no. 6, pp. 610-621, 1973.
- [12] A. Hewgill and B. J. Ross, "Procedural 3D texture synthesis using genetic programming," *Computers and Graphics*, vol. 28, no. 4, pp. 569-584, 2004.
- [13] A. K. Jain and K. Karu, "Learning texture discrimination masks," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 18, no. 2, pp. 195-205, 1996.
- [14] J. K. Kishore, L. M. Patnaik, V. Mani, and V. K. Agrawal, "Application of genetic programming for multiclass pattern classification," *IEEE Transactions on Evolutionary Computation*, vol. 4, no. 3, pp. 242-258, 2000.
- [15] B. Lam and V. Ciesielski, "Discovery of human competitive image texture feature extraction programs using genetic programming," in *Proceedings of the Conference on Genetic and Evolutionary Computation (GECCO '04)*, K. Deb, R. Poli, W. Banzhaf, et al., Eds., vol. 2, pp. 1114-1125, Springer, Seattle, Wash, USA, June 2004.

- [16] K. Laws, "Rapid texture identification," in *Image Processing for Missile Guidance*, vol. 238 of *Proceedings of SPIE*, pp. 376–380, San Diego, Calif, USA, July-August 1980.
- [17] S. Livens, P. Scheunders, G. van de Wouwer, and D. van Dyck, "Wavelets for texture analysis, an overview," in *Proceedings of 6th International Conference on Image Processing and Its Applications*, vol. 2, pp. 581–585, Dublin, Ireland, July 1997.
- [18] T. Loveard and V. Ciesielski, "Representing classification problems in genetic programming," in *Proceedings of the IEEE Conference on Evolutionary Computation (ICEC '01)*, J. H. Kim, Ed., vol. 2, pp. 1070–1077, Seoul, South Korea, May 2001.
- [19] C. Palm, "Color texture classification by integrative co-occurrence matrices," *Pattern Recognition*, vol. 37, no. 5, pp. 965–976, 2004.
- [20] N. Paragios and R. Deriche, "Geodesic active regions and level set methods for supervised texture segmentation," *International Journal of Computer Vision*, vol. 46, no. 3, pp. 223–247, 2002.
- [21] R. Poli, "Genetic programming for feature detection and image segmentation," in *Selected Papers from AISB Workshop on Evolutionary Computing*, pp. 110–125, Springer, New York, NY, USA, 1996.
- [22] B. J. Ross, A. G. Gualtieri, F. Fueten, and P. Budkewitsch, "Hyperspectral image analysis using genetic programming," in *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO '02)*, pp. 1196–1203, Morgan Kaufmann, New York, NY, USA, July 2002.
- [23] A. Song, *Texture classification: a genetic programming approach*, Ph.D. thesis, Department of Computer Science, RMIT, Melbourne, Australia, 2003, <http://www.cs.rmit.edu.au/asong/thesis.pdf>.
- [24] A. Song and V. Ciesielski, "Texture classifiers generated by genetic programming," in *Proceedings of the Congress on Evolutionary Computation*, X. Yao, Ed., vol. 1, pp. 243–248, Honolulu, Hawaii, USA, May 2002.
- [25] A. Song, T. Loveard, and V. Ciesielski, "Towards genetic programming for texture classification," in *Proceedings of the 14th Australian Joint Conference on Artificial Intelligence*, M. Sumpter, D. Corbett, and M. Brooks, Eds., vol. 2256 of *LNAI*, pp. 461–472, Springer, Berlin, Germany, December 2001.
- [26] I. Stainvas and D. Lowe, "A generative probabilistic oriented wavelet model for texture segmentation," *Neural Processing Letters*, vol. 17, no. 3, pp. 217–238, 2003.
- [27] T. Sziranyi and M. Csapodi, "Texture classification and segmentation by cellular neural networks using genetic learning," *Computer Vision and Image Understanding*, vol. 71, no. 3, pp. 255–270, 1998.
- [28] M. Tuceryan and A. K. Jain, "Texture analysis," in *Handbook of Pattern Recognition and Computer Vision*, C. H. Chen, L. F. Pau, and P. S. P. Wang, Eds., chapter 2, pp. 235–276, World Scientific, Singapore, 1993.
- [29] P. Wagner, "Texture analysis," in *Handbook of Computer Vision and Applications*, B. Jahne, H. Haussecker, and P. Geissler, Eds., vol. 2, chapter 12, pp. 275–308, Academic Press, San Diego, Calif, USA, 1999.
- [30] I. H. Witten and E. Frank, *Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations*, Morgan Kaufmann, San Francisco, Calif, USA, 2000.
- [31] S.-C. Zhu, C.-E. Guo, Y. Wang, and Z. Xu, "What are textons?" *International Journal of Computer Vision*, vol. 62, no. 1-2, pp. 121–143, 2005.

Vic Ciesielski: School of Computer Science and Information Technology, RMIT University, Melbourne, VIC 3001, Australia

Email: vic.ciesielski@rmit.edu.au

Andy Song: School of Computer Science and Information Technology, RMIT University, Melbourne, VIC 3001, Australia

Email: andy.song@rmit.edu.au

Brian Lam: School of Computer Science and Information Technology, RMIT University, Melbourne, VIC 3001, Australia

Email: brian.lam@unitedgroup ltd.com

11

A framework for the adaptation of image operators

Mario Köppen and Raul Vicente-Garcia

11.1. Introduction

The automated tuning of image processing operations is an important task for the improvement of the robustness, reliability, and versatility of image processing systems. Nearly every approach in this field is based on the classical image processing chain (IPC), which consists of a sequence of single image processing operations steps that are designed independently. Mostly notable steps here are the image acquisition, the computation of features and their classification. Other steps that might extend the processing chain are image enhancement, region-of-interest specification or image segmentation before feature computation, feature selection or feature transformation before classification, and semantic processing of images classes or object detection following the classification. Among many textbooks about this field, see especially [15] for an excellent introduction and motivation.

The versatility of the IPC scheme is usually achieved by means of a training procedure (see Figure 11.1). Given a training set (either labeled data supplied by the user, or unlabeled ones in the so-called unsupervised learning mode), the training scheme may modify some of the internal settings of the steps in the IPC in order to achieve the best mapping function from images to classes. A remarkable issue here is that the single steps in the processing flow consider their input usually as immutable: the feature computation does not modify the image acquisition procedure, the classification does not influence the manner in which the features were computed. The training overcomes this drawback by being given some influence on these settings, like modification of some parameters of the feature computation, or modification of internal parameters of the classification. However, from the fact that each step was designed independently, it follows that training can always be broken into parts, which tune a single step only.

From this description, the most important drawback of the IPC approach becomes obvious: the IPC as a whole cannot perform better than its worst configured part.

A print edition of this book can be purchased at

<http://www.hindawi.com/spc.8.html>

<http://www.amazon.com/dp/9774540018>

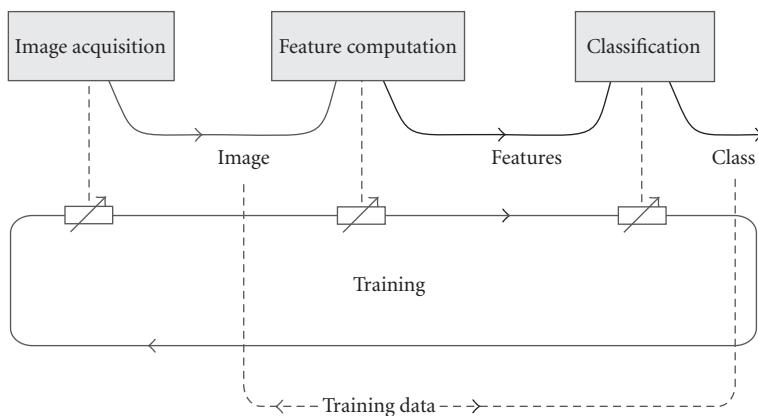


FIGURE 11.1. Typical processing flow for the training of an image processing system.

However, another problem with the IPC approach is not that obvious: each free parameter of an IPC gives an additional dimension of the search space of the optimization problem that corresponds to the training. Thus, the search space seems to become quite large. However, a simple quantitative investigation of the case gives, that the search space, if given in terms of the represented image (and not the IPC free parameters) is apparently greater, since it is given by a mapping. The number of possible mappings of n variables with m possible values onto k values is $k^{(m^n)}$. In fact, the percentage of possible image processing operations that can be represented by an IPC with respect to the total number of *all* possible image processing operations is nearly zero, even if the domain of the latter ones is highly restricted.

When soft computing techniques like genetic algorithms, genetic programming or neural networks are to be applied to the design of image processing operations, the question of how image processing operations can be adequately represented for this purpose becomes very important.

The evolutionarily designed primary visual system of higher mammals gives an important hint on this issue. Here, for example, the model of the Boundary Contour System/Feature Contour System [5] introduced two independent pathways in the cortical processing of (at least) static images, which are perceived by the retina and cortical cells.

The point of interest in the context mentioned afore is the use of several pathways instead of a single one (i.e., an IPC). Within the final fusion of the processing results of each pathway, a mapping can be included, which dramatically increases the number of representable operations. If the mapping can be specified independently, it can be expected that the task of image processing operation tuning can be solved much more effectively.

Once having the general idea of such n -dimensional frameworks, the main body of this chapter is considering a realization of such a framework. It comes out that a basic generic algorithm, originating in the image processing discipline of

mathematical morphology, the so-called 2D lookup, provides all that is needed to design such a framework.

For adapting the framework to the solution of a given image processing task, genetic programming (GP) will be used. The task here is to evolve the operations (“pathways”) that give the input for the mapping in the framework. Using GP for adapting the IPC has been studied a few times in the past. The seminal work of Tackett [20] used GP to derive efficient feature computations, and the application was the recognition of targets. Harris and Buxton proposed the use of GP for deriving edge detectors in 1D signals [6], while Poli studied the more general use of GP in several image processing tasks like image enhancement, image filtering, and feature classification [14]. A specialization to the detection of objects or regions-of-interest was presented by Bhanu and Lin [2, 3]. All these approaches followed the general idea to synthesize more complex operators from simple ones and showed the efficiency of such approaches. Using rules instead was considered by Stanhope and Daida in [18]. A recent work by Lam and Ciesielski introduced the computation of translation invariant features that are evolved by GP and evaluated by a clustering procedure [12].

The motivation to use a 2D-Lookup algorithm as internal mapping in an IPC was firstly inspired by the successful application of a 2D-Lookup for the segmentation of background texture in images of bank checks [4]. Later on, a refined version was presented by Köppen et al. in [8, 9]. The application of the presented framework to an industrial collagen-sheet inspection system can be found in [13], and its application to texture detection (with the intention to select appropriate image region for digital watermarking) can be found in [7].

In this contribution, the 2D frameworks based on this general idea and means for its extension will be presented. In Section 11.2 the concept of an n -dimensional framework will be presented and discussed, followed by Section 11.3 that introduces the 2D-Lookup algorithm for realizing such a framework. Details of the framework are presented in Section 11.4, and the possible extensions are discussed in Section 11.5. After the provision of some results for the application of the framework in Section 11.6, the chapter concludes with a short summary and the reference.

11.2. Multidimensional frameworks

In this section, a rough estimation should be made about the dimensionalities of the search problem involved in an optimization approach to IPC configuration. The basic assumption is that the result of the IPC should resemble a given goal image as good as possible, or that some properties are fulfilled by the result image. From this, the configuration of an IPC comes out to be an optimization problem.

Consider Figure 11.2, where a simple unit IPC is given. It is assumed that the computations are restricted within a 3×3 neighborhood at each pixel. All grayvalues at image pixel locations are assumed to be values between 0 and 255. The result of the operation should be a binary one, that is, the computations give a value of either 0 or 1. Then, such a unit IPC can be considered as a mapping

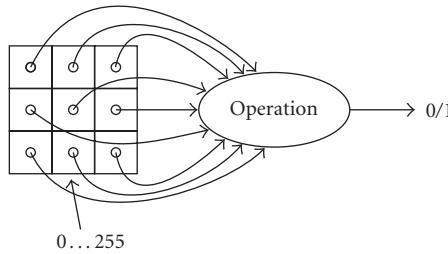


FIGURE 11.2. The unit IPC which maps nine grayvalues onto two.

$f : \{0, \dots, 255\}^9 \rightarrow \{0, 1\}$ from nine grayvalues onto the set of binary values $\{0, 1\}$. If there is a quality function q that assigns a quality value to each mapping f , a mapping f_{opt} is searched, for which its quality value becomes optimal. However, since for the mapping f there are 256^9 function values to specify, each of which can either be 0 or 1, there are $2^{(256^9)}$ possible mappings for the unit IPC. This is the number of elements of the search space, too.

As an example, consider the class of convolution operations. A weighted mask is given by the mapping of an index set M (the mask) into the set of weights, like

$$M_w = \begin{array}{|c|c|c|} \hline & w_{(-1,-1)} & w_{(0,-1)} & w_{(1,-1)} \\ \hline & w_{(-1,0)} & w_{(0,0)} & w_{(1,0)} \\ \hline & w_{(-1,1)} & w_{(0,1)} & w_{(1,1)} \\ \hline \end{array} \quad (11.1)$$

and assuming here that $w_{ij} \in \{0, \dots, 255\}$. Then, convolving the image I at position (x, y) with the weighted mask M_w can be written as

$$R(x, y) = \sum_{(i,j) \in M} w(i, j)I(x + i, y + j). \quad (11.2)$$

By thresholding the result with a value ϑ , a binary image is obtained. For this unit IPC with nine parameters, there are 256^9 possible choices for the parameters. Compared with the number of search space elements $2^{(256^9)}$, this is only a marginal amount of representable operations. The number p of parameters with a domain $\{0, \dots, 255\}$ that would be needed to cover the search space completely is $256^9/8$. This follows from equating $2^{(256^9)} = 256^p = 2^p$.

A serious problem arises from this consideration. If adaptive techniques like soft computing methods should be applied to such an image processing problem, the search space is much too big to be covered by the search method. There seems to be no way to represent an arbitrary mapping of the kind of mappings used in image processing operations.

The number of represented operations could be dramatically increased, if a mapping would be involved into the IPC operations. This leads to the definition

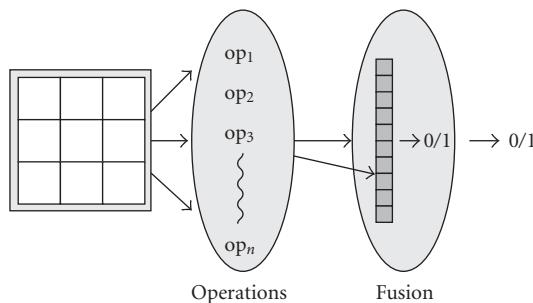


FIGURE 11.3. An n -dimensional framework, which decomposes an IPC into n operations performed in parallel and a final fusion of the operation results. The fusion can be based on a mapping, thus heavily increasing the number of representable operations.

of n -dimensional frameworks. An n -dimensional framework is a decomposition of the processing flow into n parallel parts op_1 to op_n and a final fusion procedure (see Figure 11.3). Each single operation is applied onto the original image, and then, the n result images are fused by an appropriate algorithm. If the fusion is specified by a mapping of n values out of a set of m values each onto the set $\{0, 1\}$, the framework, as seen from the “outside,” serves as a unit IPC of the kind given above. There are $2^{(m^n)}$ mappings specified. If m is set to 256 and n to 9, we exactly meet the requirements of the unit IPC.

This is the key idea of n -dimensional frameworks: they allow for its adaptation as a whole by adapting the parameters of n operations (each of which could be an IPC itself), thereby sampling the search space to a much more larger degree as can be achieved by setting a number of internal parameters only.

The question is, of course, if there is a fusion algorithm that really supports the adaptation of image processing operators. The answer is positive, and in the next section, the 2D-Lookup algorithm will be identified as such a fusion procedure.

11.3. 2D-Lookup algorithm

The 2D-Lookup algorithm stems from mathematical morphology [16, 17]. It was primarily intended for the segmentation of color images. However, the algorithm can be generalized for using on grayvalue images as well.

For applying the 2D-Lookup algorithm, two input images g_1 and g_2 of same size and number of bits per pixel are required. These two images can be the result of applying two image operators onto a single input image, or by extracting color channels from a color representation of an input image. The nature of the image operators or channel selections itself does not matter for the application of the 2D lookup. The other component of the algorithm is a matrix of dimension $N_g \times N_g$ (with N_g being the number of different grayvalues in the two input images) and having entries from a set of labels, with preference to the same grayvalue range as the input images.

```

for x=0 to img_width-1 do
begin
  for y=0 to img_height-1 do
  begin
    g1 = g1(x,y)
    g2 = g2(x,y)
    out(x,y) = l(g1,g2)
  end y
end x

```

ALGORITHM 11.1

The 2D-Lookup algorithm goes over all common positions of the two-operation images. For each position, the two pixel values at this position in the images g_1 and g_2 are used as indices for looking up the 2D-Lookup matrix. The matrix element, which is found there, is used as pixel value for this position of the result image. If the matrix is bi-valued, the resulting image is a binary image.

Let I_1 and I_2 be two grayvalue images, defined by their image functions g_1 and g_2 over their common domain $P \subseteq N \times N$:

$$\begin{aligned} g_1 : P &\longrightarrow \{0, \dots, g_{\max}\}, \\ g_2 : P &\longrightarrow \{0, \dots, g_{\max}\}. \end{aligned} \quad (11.3)$$

The 2D-Lookup matrix is also given as an image function l , but its domain is not the set of all image positions but the set of tuples of possible grayvalue pairs $\{0, \dots, g_{\max}\} \times \{0, \dots, g_{\max}\}$,

$$l : \{0, \dots, g_{\max}\} \times \{0, \dots, g_{\max}\} \longrightarrow S \subseteq \{0, \dots, g_{\max}\}. \quad (11.4)$$

Then, the resulting image function is given by

$$\begin{aligned} r : P &\longrightarrow S, \\ r(x, y) &= l(g_1(x, y), g_2(x, y)). \end{aligned} \quad (11.5)$$

In standard applications, every grayvalue is coded by eight bit, resulting in a maximum grayvalue of 255. Also, the domain of the image function is a rectangle. In this case, the 2D-Lookup is performed by the pseudocode shown in Algorithm 11.1.

To give a simple example for the 2D-Lookup procedure, $g_{\max} = 3$ is assumed in the following. Let

$$g_1 : \begin{array}{|c|c|c|} \hline 0 & 1 & 2 \\ \hline 0 & 3 & 3 \\ \hline \end{array}, \quad g_2 : \begin{array}{|c|c|c|} \hline 2 & 3 & 1 \\ \hline 2 & 3 & 2 \\ \hline \end{array} \quad (11.6)$$

be the two input images and let the 2D-Lookup matrix be given by

g_1	0	1	2	3
g_2	0	0	1	1
0	0	0	1	1
1	0	1	2	2
2	1	2	3	3
3	2	3	3	2

(11.7)

Then, the resulting image is

$$r : \begin{bmatrix} l(0,2) & l(1,3) & l(2,1) \\ l(0,2) & l(3,3) & l(3,2) \end{bmatrix} = \begin{bmatrix} 1 & 3 & 2 \\ 1 & 2 & 3 \end{bmatrix} \quad (11.8)$$

In the following, the 2D-Lookup matrix will only contain the two entries: Black (0) and White (1).

A typical base for the matrix can be the so-called 2D histogram. Using the former notation, the 2D histogram of two images g_1 and g_2 is a mapping

$$H(g_a, g_b) = \sum_{(x,y) \in P} \delta(g_1(x,y), g_a) \delta(g_2(x,y), g_b) \quad (11.9)$$

with $\delta(a, b)$ being 1 for $a = b$ and 0 otherwise. The entry of H at position of a grayvalue pair (g_a, g_b) contains the number of pixel positions (x, y) where image g_1 has grayvalue g_a and image g_2 has grayvalue g_b . Using a normalization factor (like the maximum value in H), the 2D histogram can be given as an image and used as a 2D-Lookup matrix.

Figure 11.4 illustrates the relations between 2D histogram and 2D lookup by means of the Lena image. The subimage to the middle right is the 2D histogram of the Lena image that was obtained from the red and green channels of the Lena image (taken as grayvalue images). There are some obvious clusters in this histogram that give rise to a labeling as shown in the figure. The five binary images were obtained by using a 2D-Lookup matrix image each, having all positions of the segment number i set to black, and white otherwise. So, it can be seen that the binary image generated by setting all points of label 3 to black (lower-left subimage) basically covers the mirror structure in the image background. Since the projection of label 3 into both axis directions of the label image is crossing label 2, it is not possible to extract this mirror structure from the red or green channel image of the Lena image alone. More precisely: if one tries to extract the grayvalue range spanned by the positions of the mirror structure, in either case (red or green) also

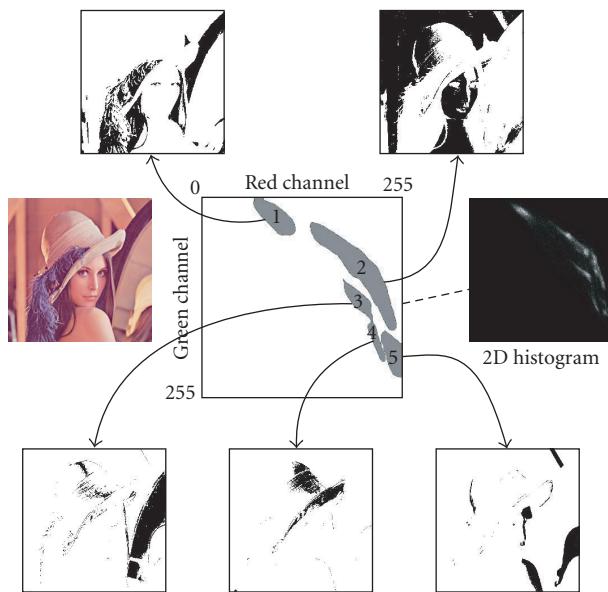


FIGURE 11.4. Various segmentations of Lena image based on labeling of the 2D histogram of red and green channels.

some other structures are always selected as well that do not belong to the mirror structure. The projections of the 2D histogram gives a clear indication for this. The 2D-Lookup algorithm is able to separate the mirror structure directly. From this simple example, the potential of the 2D-Lookup algorithm for segmentation tasks can be seen.

11.4. 2D-Lookup-based framework

The 2D-Lookup-based framework (see Figure 11.5) is composed of (user-given) original image, filter generator, operation images 1 and 2, result image, (user-supplied) goal image, 2D-Lookup matrix, comparing unit, and filter generation signal.

The framework can be thought of as being composed of three (overlapping) layers.

- (1) The instruction layer, which consists of the user-supplied parts of the framework: original image and goal image.
- (2) The algorithm layer performs the actual 2D-Lookup, once all of its components (original image, operation 1, operation 2, and 2D-Lookup matrix) are given.
- (3) The adaptation layer contains all adaptable components of the framework (operation 1, operation 2, 2D-Lookup matrix) and additional components for performing the adaptation (comparison unit, filter generator).

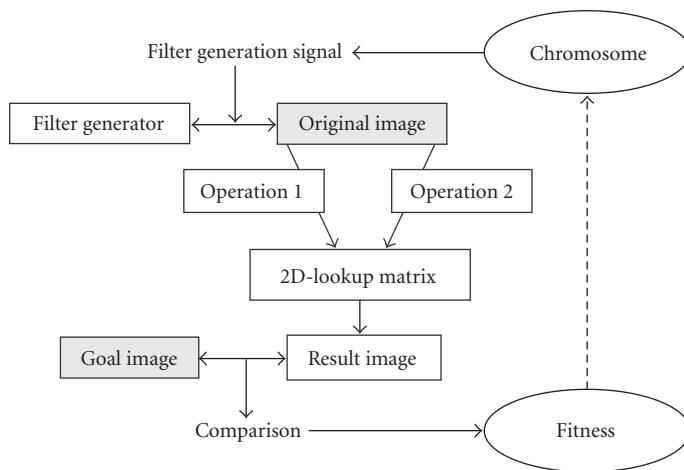


FIGURE 11.5. 2D-Lookup-based framework: overview.

For the instruction layer, the user interface has been designed as simple as possible. The user instructs the framework by manually drawing a (binary) goal image that provides the wanted segmentation of the original image into foreground and background. In this image, all pixels of the background segment are set to White and all pixels of the foreground (e.g., the location of a texture fault, or the handwriting on a textured bankcheck background) are set to Black. No special texture model has to be known by the user. There are no further requirements for the goal image.

For making the framework the subject of an evolutionary adaptation, several aspects have to be considered in more detail.

- (1) Fitness function: assuming the operation images and the 2D-Lookup matrix to be given, the result of 2D lookup has to be compared with the user-supplied binary goal image (indicated as “comparison” in Figure 11.5). This is achieved by a fitness function that measures the degree of spatial correspondence between two binary images.
- (2) Derivation of 2D-Lookup matrix: once the two-operation images are known, it can be expected that the 2D-Lookup matrix can be adapted in order to give the best fitness to the goal image as result of the 2D lookup. Later on (see Section 11.4.2) we will provide exactly such a procedure, making the 2D-Lookup matrix a function of the operation images 1 and 2 and the goal image alone, and without the need of any further adaptation.
- (3) Having the fitness function and the method to derive the 2D-Lookup matrix, it only remains to specify the two-operation images. Here, the task can be handled by an evolutionary procedure. In summary, we decided to use the representation of operations as expression trees, and using genetic programming in order to derive optimal image operations.

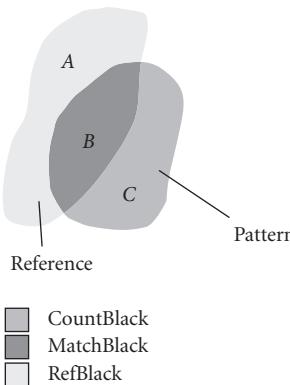


FIGURE 11.6. Terms for fitness evaluation.

In the following sections, these necessary specifications of the framework will be presented.

11.4.1. Fitness function

In order to compare the output image of the 2D Lookup with the goal image, a quality function has to be designed for the comparison of two binary images. First, the definition of this fitness function will be given, then it will be discussed.

Consider Figure 11.6, where two sets are shown, the reference set of the goal image and the pattern set of the result image. The reference set of the goal image is the set of all black pixels in the goal image that is given by the user. The pattern set of the result image is the set of all black pixels of the result image.

Therein, countBlack is the number of black pixels in the result image ($B + C$), matchBlack is the number of black pixels of the result image, which are also black in the goal image (B), and refBlack is the number of black pixels of the goal image ($A + B$). Then, the following ratios can be computed:

$$r_1 = \frac{\text{matchBlack}}{\text{refBlack}}, \quad (11.10)$$

where r_1 is the sensitivity, or amount of reference pixels matched by the pattern,

$$r_2 = 1.0 - \frac{\text{countBlack} - \text{matchBlack}}{N - \text{refBlack}}, \quad (11.11)$$

where r_2 is the specificity, or amount of correct white pixels set in the result image (N is the total number of image pixels), and

$$r_3 = \frac{\text{matchBlack}}{\text{countBlack}}, \quad (11.12)$$

where r_3 is the positive predictivity, or percentage of matching pixels of the result image. In case of empty reference or pattern sets (white images), r_1 and r_3 are set to 0, respectively.

The multiple objective here is to increase these measures simultaneously. After performing some experiments with the framework, it was decided to use the following weighted sum of these three objectives as fitness measure:

$$f_{\text{reference}}(\text{pattern}) = 0.1r_1 + 0.5r_2 + 0.4r_3. \quad (11.13)$$

This fitness measure has the following properties.

- (1) It counts better for patterns that are subsets of the reference. Subsets obtain a fitness value of at least 0.9, since in this case specificity r_2 and positive predictivity r_3 both take the value 1.
- (2) It counts better for patterns that are subsets of the reference, and which are supersets of other patterns that are also subsets of the reference (supersets of patterns have a larger value for sensitivity r_1 than the pattern, and again, $r_2 = r_3 = 1$).
- (3) A white image as pattern gives a fitness of 0.5 (only specificity $r_2 \neq 0$), therewith refusing to assign a good fitness value to the empty subset of the reference.

These properties make this fitness measure useful for heuristic search procedures. Initially, higher fitness values can be obtained by increasing the higher weighted objective first. In our case this means that specificity r_2 , weighted with 0.5, is improved first. In other words, the first subgoal of the search could be to allocate as many correct white positions as possible. Due to the slightly smaller weighting of 0.4 for the positive predictivity r_3 , the search then could continue to allocate also correct black positions of the reference, while the correct white allocations persist in the pattern. Once, by this exploration, the pattern is reduced to a subset of the reference, the only way to increase the fitness is to expand the subset towards the whole reference set. This begins, when the fitness exceeds a value of about 0.9, and exploitation starts.

11.4.2. Deriving a 2D-Lookup matrix

It was already noted that the specification of a 2D-Lookup matrix can be done without the need for a separate adaptation. To make this more clearly, we recall the dependencies between the parts of the framework. Using the notations in for the original image, $goal$ for the goal image, op_1 and op_2 for the two-operation images after applying the operators o_1 and o_2 to in , res for the result image and lt_2 for the 2D-Lookup matrix, the 2D-Lookup LU_2 can be formally written as

$$res = LU_2(o_1(in), o_2(in), lt_2) = LU_2(op_1, op_2, lt_2), \quad (11.14)$$

and thus it can be expected that there is also an operation or algorithm REL to specify lt_2 from the other terms in (11.14):

$$lt_2 = \text{REL}(\text{op}_1, \text{op}_2, \text{res}) = \text{REL}(\text{op}_1, \text{op}_2, \text{goal}). \quad (11.15)$$

The replacement of res with goal expresses the fact that the primary intention of the adaptation is to have the relation $\text{res} = \text{goal}$ fulfilled as good as possible.

For specifying REL, we consider a family of simple 2D-Lookup matrices, where all positions but one position (a, b) are set to White (1), and the remaining position (a, b) is set to Black (0). Then, the 2D lookup will give a resulting image with all positions (x, y) set to Black, for which operation op_1 yielded pixel value a at (x, y) in op_1 and operation op_2 yielded pixel value b at (x, y) in op_2 . Usually, there will be only a few black pixels within the result image. Now, we compute the fitness measure in (11.13) taking the set of black positions in the result image as the pattern set, and the black pixels of the goal image as reference set. As it was remarked in the former section, the fitness measure will give values above 0.9, if the pattern set of black pixels lies completely within the reference set, even if this set contains only one pixel. So, a simple criterion can be derived for setting a pixel to Black or White in the 2D-Lookup matrix.

Let $l_{(a,b)}$ be a two-dimensional matrix constituted by setting only the position at (a, b) to Black (0) and all others to White (1), and let $\text{res}_{(a,b)}$ be the result of the 2D lookup with the operation images op_1 and op_2 and this matrix $l_{(a,b)}$. Then we use for REL:

$$lt_2(a, b) = \begin{cases} \text{Black (0)} & \text{if } f_{\text{goal}}(\text{res}_{(a,b)}) > 0.88, \\ \text{White (1)} & \text{otherwise.} \end{cases} \quad (11.16)$$

It can be seen that this procedure REL only requires the operation images op_1 and op_2 , and the goal image goal . In case there are no black pixels in $\text{res}_{(a,b)}$ at all, $lt_2(a, b)$ is set to Gray (0.5), which stands for positions within the 2D-Lookup matrix, whose pixel value pairs do never occur within the operation images op_1 and op_2 at the same location. The value 0.88 has been chosen instead of 0.9 to tolerate a few black pixels in $\text{res}_{(a,b)}$ to be out of the goal set.

Figure 11.7 shows the result of the algorithm REL for the derivation of a suitable 2D-Lookup matrix for two example operation images. This procedure, which resembles a relaxation procedure, gives a quasioptimal 2D-Lookup matrix for given operation images op_1 and op_2 .

By this specification, the algorithm looks rather time consuming. However, by doing some book keeping, and using the fact that only points will be set to Black in the 2D-Lookup matrix, for which the corresponding grayvalue pairs in the operation images have at least one position that is also set black in the goal image, the processing time can be remarkably reduced. In the pseudocode shown in Algorithm 11.2, the 2D-Lookup matrix is derived as a grayvalue image $1t$ of size 256×256 and with Black positions set to grayvalue 0, White positions to 255, and Gray positions to 127.

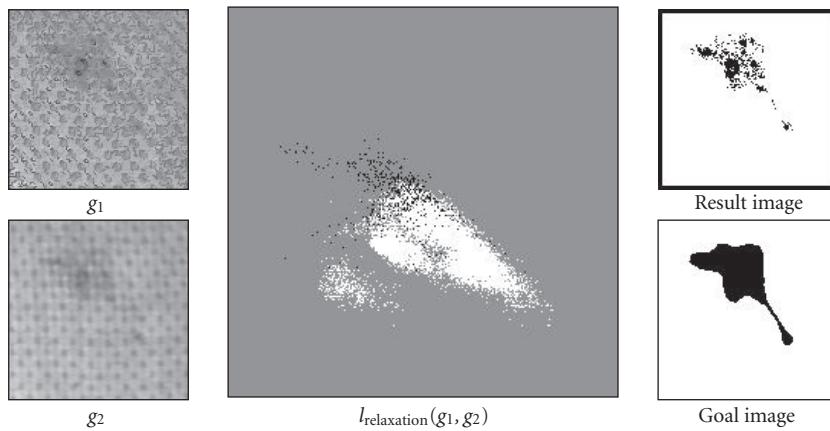


FIGURE 11.7. In the middle, the result of applying algorithm REL to two-operation images (left) and given goal image (bottom right) can be seen, and top right is the result of the application of this 2D-Lookup matrix to the two-operation images.

```

Algorithm REL:
Input: op1, op2 - operation images; goal - goal image
Output: lt - grayvalue image of size 256x256
----- 
Init:
- set all pixels of lt to 127
- compute refBlack from goal

Algorithm:
for all pixel positions (i, j) with goal (i, j) = 0
begin
    g1 := op1(i, j), g2 := op2(i, j)
    if lt (g1, g2) = 127 then
        countBlack := 0; matchBlack := 0;
        for all pixel positions (k, l) in op1
            if op1(k, l) = g1 and op2(k, l) = g2 then
                countBlack++;
                if goal (k, l) = 0 then matchBlack++; end if
            endif
            compute f = f (countBlack, matchBlack, refBlack)
            if f >= 0.88 then lt (i, j) := 0;
            else lt (i, j) := 255;
            endif
        end for all (k, l)
    endif
end for all (i, j)

```

ALGORITHM 11.2

The following section describes the manner by which the two operations needed are derived by an individual of a GP.

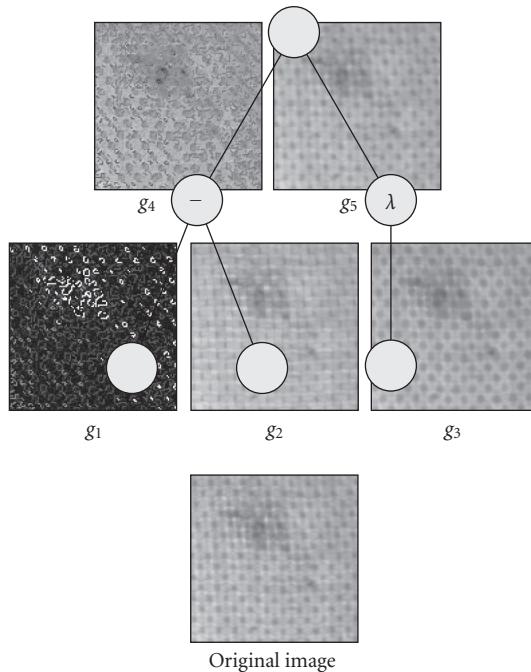


FIGURE 11.8. Example for the intermediate images that were generated for an operator tree with operation images 1 and 2 on the top row.

11.4.3. Generic operator design

The operator selection is based on a tree-like representation of the image processing operations. Each node in the tree refers to a generic image processing operation out of the set squaring, square root, move, ordered weighted averaging (OWA [21]), fuzzy T-norm, and fuzzy integral [19] for nodes of arity one, and pixelwise addition, subtraction, multiplication, and T-Norm for nodes of arity two (higher-arity nodes are not used). The parameters of such a node operation are given by a *parameter structure resource*, with the same structure for all nodes. This parameter structure resource indicates an offset vector (for move operation), a weighted mask (for OWA, T-norm and fuzzy integral), a weighting vector (for OWA) and some flags and mode values. See [8] for more details on the operation specifications.

In the framework presented here, the operator selection is performed two times, to get the operation images, from which the 2D-Lookup matrix is derived. In all cases, the operation trees and the parameter structures are randomly initialized, and adapted later on by genetic programming [10, 11] as optimization procedure to gain high fitness values.

The structuring of image processing operations by the trees has been chosen in this manner for the following reasons.

- (i) The random selection of operations, which are represented by such trees, can be done in a user-driven manner that favors well-known image processing operations. Thus, a tree could be made more likely to represent operations as dilation, erosion, closing, opening, morphological gradients, Sobel operator, statistical operators, Gaussian filtering, shadow images and so forth.
- (ii) The represented operations are unlikely to give unwanted operation images, which are completely white or black.
- (iii) The employed operators are local, that is, for each position of the result image, the computed value is a function of the grayvalues of a bounded domain (containing the same position) of the original image.

The maximum arity of a node is set to two. Also, maximum tree depth was restricted to five. This was set in order to allow for the maintenance of the obtained trees, for example, for manually improving the designed filters by removing redundant branches. Processing time is kept low, too (but in the present version of the framework, processing time does not go into the fitness function itself!).

Figure 11.8 shows a typical tree constructed in this manner and applied to an input image with a fault structure on textured background (bottom subimage). Figure 11.9 gives some operation images obtained from the same original image by different randomly constructed and configured trees. These images demonstrate the variability of the generated operations, each of which enhances or surprises different image substructures, and none of which gives a trivial image operation.

11.5. Framework extensions

In this section, we will introduce two optional extensions of the framework. The next section is devoted to the aspect of obtaining filters with higher generalization ability, and then, an optional preprocessing module based on 2D lookup with the 2D histogram is presented.

11.5.1. Reconstruction of the 2D-Lookup matrix

An important question is the generalization ability of the designed filters. While they were designed for one and only one input-goal image pair, it may not be obvious how the filter performs, if they are applied to the same situation presented by another image.

The key for checking generalization ability of such a designed filter is given by the generated 2D-Lookup matrices. These matrices can be manipulated for improving the filter's generalization ability. Figure 11.10 shows some 2D-Lookup matrices, which were the result of applying the framework to various texture fault examples (the texture faults themselves are of minor interest for the following). By considering these matrices, the following can be noted.

- (i) Some matrices seem to be separable, that is, the textured background is represented by a group of compact white regions. The fault appearance (the black dots) are well separated from these regions.

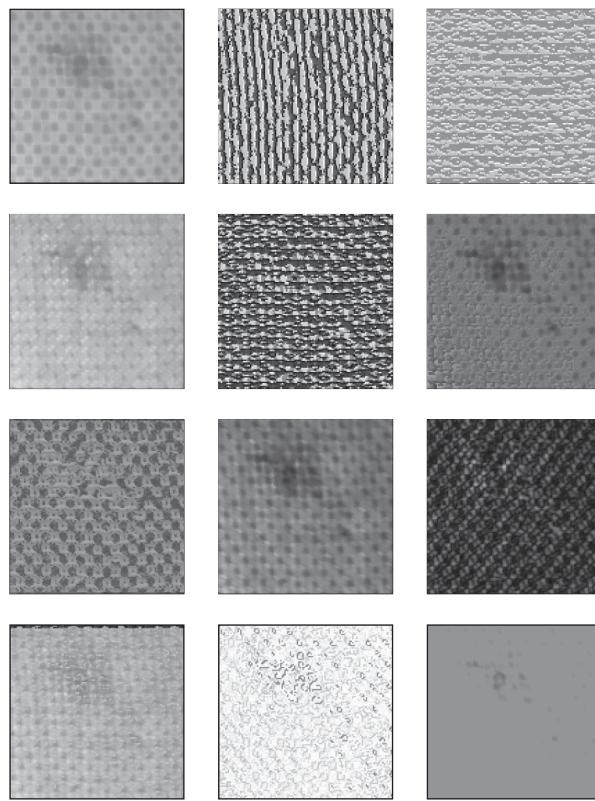


FIGURE 11.9. Result images of a random initialization of a population of image processing operators.

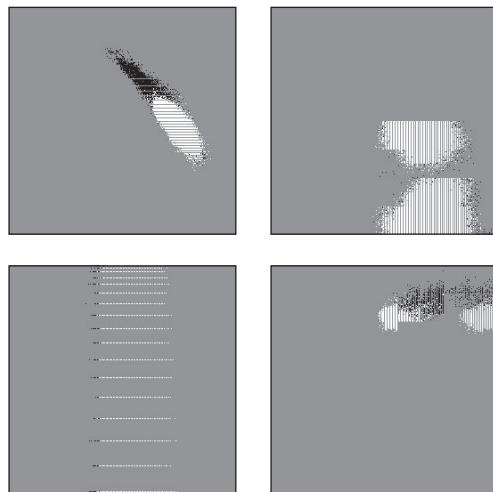


FIGURE 11.10. Example for 2D-Lookup matrices generated by the 2D-Lookup framework.

- (ii) The gray parts of the matrices represent positions, for which no special rule could be assigned, since the corresponding pair of grayvalues was not present in the operation images at the same position.
- (iii) There are noncompact, noisy regions, where black and white dots are completely admixed. This seems to be a combined effect. The grayvalues around some position, which was indicated as foreground in the goal image, could be similar (but not equal) to grayvalues around positions that were indicated as background in both operation images. This is a conflict in the assignment, as given by the goal image, and the rapid changes between black and white for closeby positions reflect such an ambiguity of an optimal assignment. The conclusion is that these filters are not able to achieve a good separation between image foreground and background.
- (iv) Some matrices are separable by a horizontal or vertical straight line (e.g., lower left example of Figure 11.10). This means that the 2D-Lookup algorithm can be simplified to thresholding op_2 or op_1 , respectively.

To summarize: compactness within the 2D-Lookup matrices is considered as main provision for filter's high generalization ability. If the matrices are manipulated in a manner, which enhances compactness of its black and white regions, the filter will perform better on newly presented images (possibly for the price of a slightly lower performance on the input image, from which the filter was designed).

But filter performance is not the only advantage of such a procedure. If it would be possible to provide a description of the compact regions within a 2D-Lookup matrix on a higher level than by its plain pixel sets, this information would allow for deriving texture models from the framework's results.

This leads to a new statement of the problem: to find out about compactness within the class of images, to which 2D-Lookup matrices belong.

Neural networks would give a suitable procedure for the segmentation of the matrix, since they attain generalization from data. In the following, an approach based on the use of the unit radial basis function network (Unit-RBF), as proposed in [1] will be given. The Unit-RBF approach is directly applicable to the problem of approximating 2D-Lookup matrix images. The derived 2D-Lookup matrix was decomposed into a black part and a white part. Since there are often fewer foreground pixels in the goal images than background pixels, the black part image was further processed by morphological closing operation (to join some isolated black dots into a single segment). Then, the Unit-RBF approach was used to reconstruct both images, rec_1 from the black part, rec_2 from the white part, and both images were pixel-wise fused into a single image by using the rules given in Table 11.1. Note that the entry 127 stands for undecidable positions, where no evidence can be obtained from the framework adaptation.

Figures 11.11 and 11.12 give some examples for the replacement of the derived 2D-Lookup matrix with the reconstructed one. It can be seen that the differences between the results are not very large, and that the reconstructed matrices have a much simpler structure.

TABLE 11.1. Rules for fusion of the two Unit-RBF images at position (x, y) .

rec ₁	rec ₂	rec
> 127	>127	max(rec ₁ , rec ₂)
>127	\leq 127	rec ₁
\leq 127	>127	127
\leq 127	\leq 127	min(rec ₁ , rec ₂)

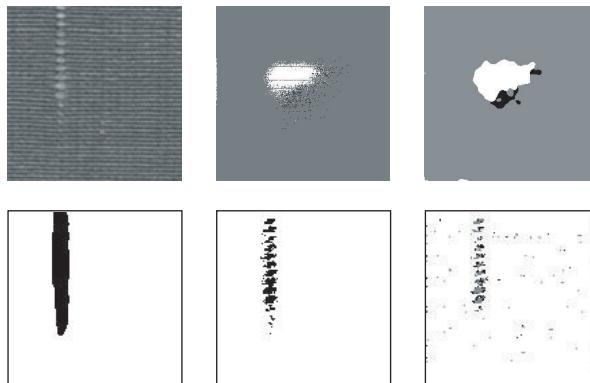


FIGURE 11.11. Reconstruction of 2D-Lookup matrix. Upper row shows the original image, the 2D-Lookup matrix as it was adapted by the 2D-Lookup-based framework, and its reconstructed version. The lower row shows the goal image, the result of 2D lookup using adapted 2D-Lookup matrix, and the result using the reconstructed matrix.

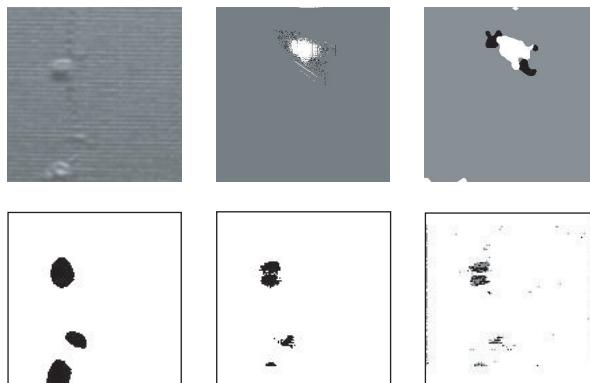


FIGURE 11.12. Another example for 2D-Lookup matrix reconstruction. The lower row shows the goal image, the result of 2D lookup using adapted 2D-Lookup matrix, and the result using the reconstructed matrix.

11.5.2. Optional preprocessing module

The framework has shown a good performance on a broad range of filtering tasks. Filtering here means the separation of image foreground and background.

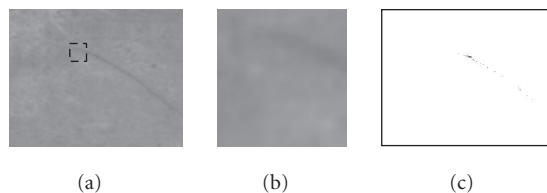


FIGURE 11.13. Low-contrast stroke processing (a) shows a stroke on a collagen sheet, (b) shows a detail enlarged, (c) gives the output of best 2D-Lookup adaptation, demonstrating the poor performance of this algorithm in this case.

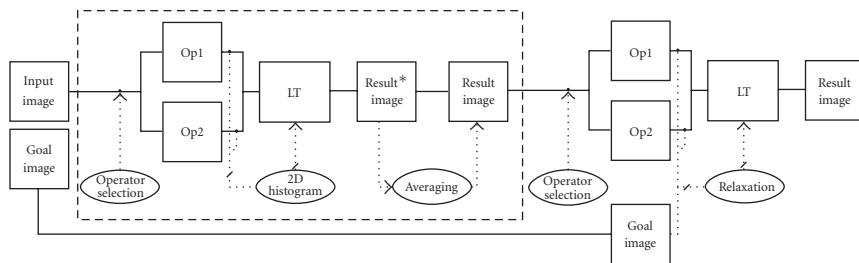


FIGURE 11.14. Extended framework for 2D-Lookup adaptation, with optional 2D histogram lookup preprocessing module.

However, it also showed some falacities. So, a rather poor performance on foreground appearance with very low contrast against the background was noted. Figure 11.13 gives an example. The stroke-like fault structure in Figure 11.13(a) (a scratch in a collagen sheet), clearly visible for a human, becomes nearly “invisible” while getting enlarged (see Figure 11.13(b)). Figure 11.13(c) shows the result of 2D-Lookup adaptation, revealing only some random dot locations of the fault.

Analyzing this problem, it came out that the framework employs local image processing operators only. Local processing here means that the domain of the image processing operations is bounded to a spatial neighborhood of each pixel. Detection of a stroke as the one in Figure 11.13 cannot be achieved by such a local processing.

This subsection presents an approach to solving such problems, by introducing an extension of the 2D-Lookup framework, referred to as 2D histogram lookup procedure. It is also based on the 2D lookup, but uses the normalized 2D histogram as 2D-Lookup matrix. The main advantage of this approach is that the mapping assigns a low grayvalue to positions with grayvalue pairs that appear infrequently in the operation images (there are fewer corresponding entries in the 2D histogram). This increases the contrast of “untypical” image structures against the background, and thus simplifies the following 2D-Lookup adaptation task.

Figure 11.14 shows the general framework for 2D-Lookup adaptation, with the optional 2D histogram lookup extension. The components will be described in

the following sections. As can be seen, the extension has nearly the same structure as the framework without extension, only the specification of the lookup matrix LT is different.

The 2D histogram lookup is the 2D-Lookup algorithm with using the normalized 2D histogram as lookup matrix (see end of Section 11.3).

For using the 2D histogram as lookup matrix, its entries have to be scaled into the range of feasible grayvalues. For usual goal image sizes, entries in the 2D histogram are seldom larger than the maximum grayvalue, so the entries themselves can be used as lookup matrix values, bounded at 255. Also, a contrast improvement by linearization is reasonable, for gaining better contrast in the result image. Linearization is the operation of making the sum-histogram of grayvalues stepwise linear. It replaces each grayvalue in the image with the relative number of grayvalues equal or below this grayvalue. Hereby, the large number of zero entries in the 2D histogram is neglected, thus starting linearization at grayvalue 1.

While gaining higher contrast by linearization, the number of different grayvalues now in the result image becomes reduced. This gives images with a cluttered appearance of the 2D-Lookup matrices in the following 2D-Lookup adaptation step. This effect can be reduced by using a Gaussian smoothing operator of size 3 on the linearized image.

The framework for 2D-Lookup adaptation was extended in order to enable the processing of low-contrast foregrounds. Figure 11.15 gives a complete example for such an adapted filtering. This example demonstrates several things.

- (i) The final result image in the lower left is much more similar to the goal image than in Figure 11.13(c).
- (ii) The result of the 2D histogram lookup preprocessing step has increased the contrast of the stroke against the background. This allows for the following 2D-Lookup adaptation step to achieve that better performance.
- (iii) The operation trees are not “smart” in the sense that parts of them may do not have much influence on the operation result of the full tree. So, the left wing of the operation 1 of the 2D lookup (the tree left below in Figure 11.15) basically produces a gray image, from which a shifted version of the preprocessed image is subtracted. Such parts can be removed in a manual redesign phase of the filters after adaptation (especially when they involve computationally more expensive node operations like the fuzzy integral).
- (iv) The 2D lookup for this example is basically a lookup with the cooccurrence matrix of the preprocessed image.
- (v) The essential part of this filter lies in the operation 2 of the preprocessing (upper right tree): the 2D histogram, with its line-like structure, forks for some higher grayvalue pairs, thus stimulating a separation of image parts into foreground and background.

Figure 11.16 shows another example, without the intermediate results. This is a so-called “vibrating knife” fault that may occur in collagen slicing. The extended framework shows a good performance here as well.

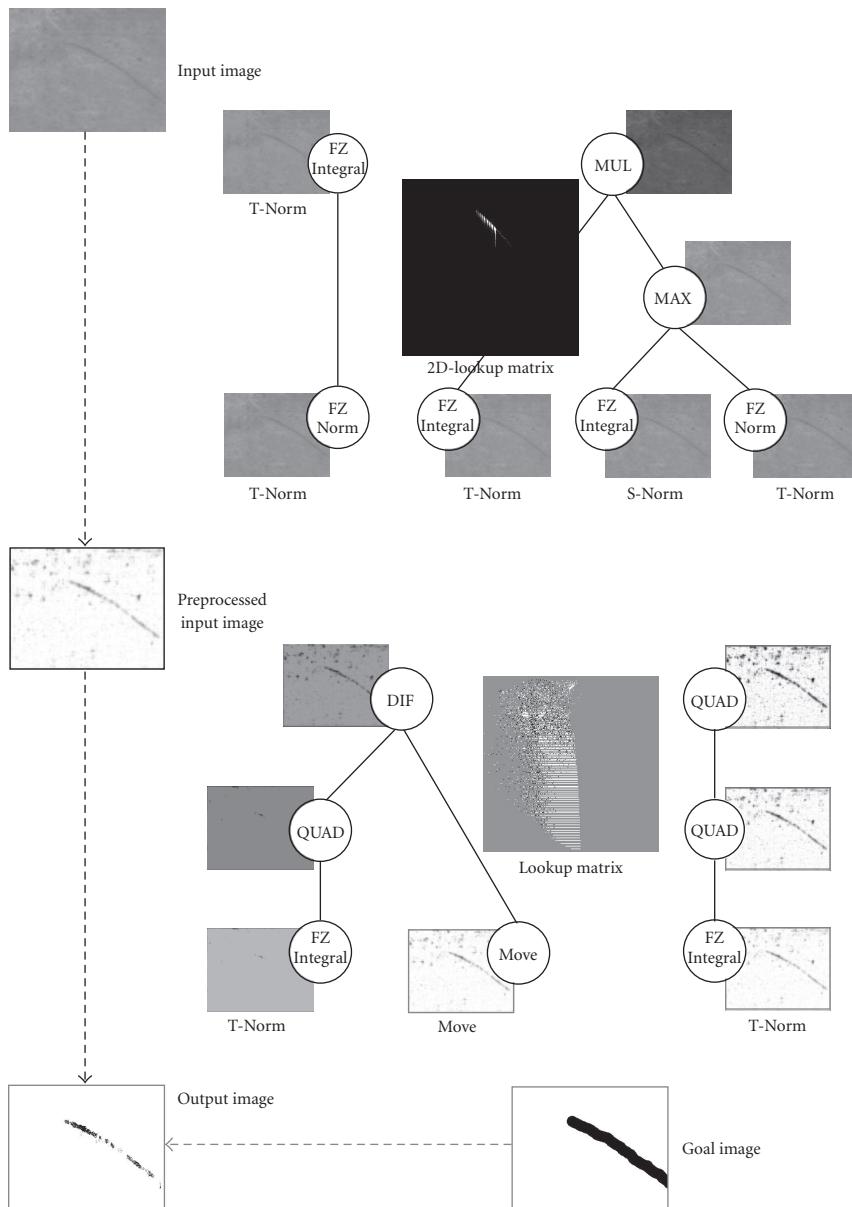


FIGURE 11.15. Complete example for the extended framework adapted to the low-contrast stroke in Figure 11.1.

11.6. Some results

In this section, some results of the application of the presented framework to some texture analysis problem are presented. For the genetic programming, setting was

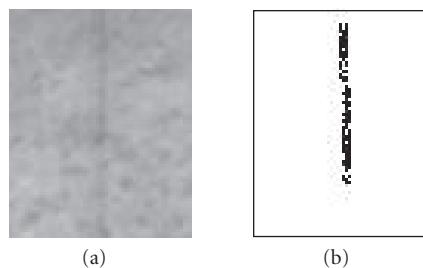


FIGURE 11.16. Result of the extended framework for vibrating-knife fault.

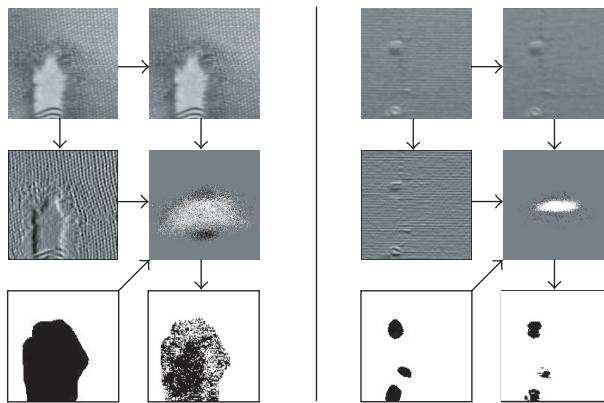


FIGURE 11.17. Application of the framework to textile fault problems.

standard: 40 trees were used in each generation, and standard tournament crossover was the only genetic operator used in each generation to produce 80 children nodes. The size of initial trees was restricted to have at least three nodes and to have not more than 9 nodes. The evolution ran until there was no diversity anymore in the population, which usually happened after about ten generations.

In Figures 11.17–11.20 the subimage order of each block is: upper-left subimage is the original image, lower-left is the user-given goal image. Right and below the original image are the two-operation images that are the result of the application of the evolutionary adapted trees to the original image, and that are the input operands for 2D-Lookup algorithm. The arrows from the operation images point to the used 2D-Lookup matrix (also indicated as a function of the two-operation images and the goal image), and lower right is the obtained result image.

11.7. Summary

A framework was presented, which allows for the design of texture filters for fault detection (two class problem). The framework is based on the 2D-Lookup algorithm, where two filter output images are used as input.

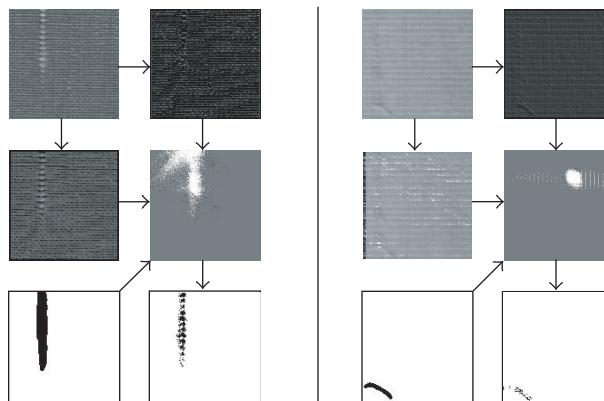


FIGURE 11.18. Application of the framework to further textile fault problems.

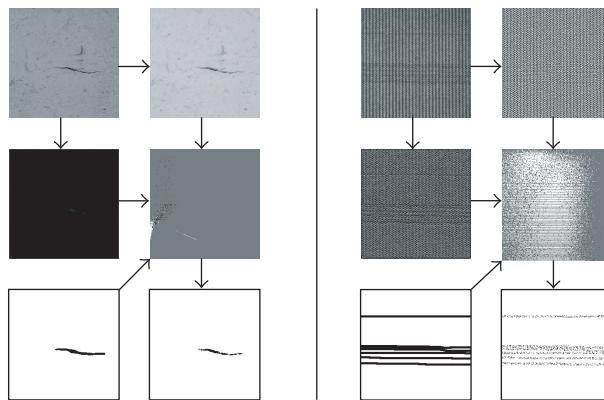


FIGURE 11.19. Application of the framework to floor pattern fault problems.

The approach can be applied to a large class of texture analysis problems. The results, obtained without “human intervention,” are ready-to-use texture filters. Also, they can be tuned in order to obtain even more better results, or combined in a superposed inspection system. The following are our experiences during the use of the system.

- (i) The framework was able to design texture filters with good or very good performance.
- (ii) The goal image matched the fault region quite satisfactorily.
- (iii) Bordering regions should be neglected for fitness evaluation.
- (iv) The framework was able to design filters for the detection of noncompact fault regions and fault regions with varying appearance.
- (v) The designed filters may be subjected to further improvements by the user.

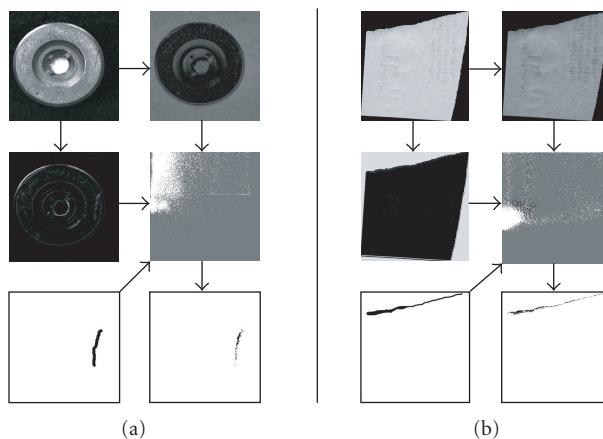


FIGURE 11.20. (a) application of the framework to cast fracture problem and (b) extraction of sheering area on torn paper piece.

Improvements of the whole architecture were considered as well: one is based on an evaluation of the 2D-Lookup matrix by neural networks in order to get a more comprehensive solution for a given texture filtering problem, the other for extending the application scope to low-contrast texture fault processing, that is, faults which are hard to separate from the background texture. The second extension of the framework is a two-stage one, based on 2D histogram lookup and consecutive 2D-Lookup adaptation.

Bibliography

- [1] P. G. Anderson, "The unit RBF network: experiments and preliminary results," in *Proceedings of the International ICSC/IFAC Symposium on Neural Computation (NC '98)*, M. Heiss, Ed., pp. 292–297, International Computer Science Conventions, Vienna, Austria, September 1998.
- [2] B. Bhanu and Y. Lin, "Learning composite operators for object detection," in *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO '02)*, pp. 1003–1010, New York, NY, USA, July 2002.
- [3] B. Bhanu and Y. Lin, "Object detection in multi-modal images using genetic programming," *Applied Soft Computing Journal*, vol. 4, no. 2, pp. 175–201, 2004.
- [4] K. Franke and M. Köppen, "Towards an universal approach to background removal in images of bankchecks," in *Proceedings of the 6th International Workshop on Frontiers in Handwriting Recognition (IWFHR '98)*, pp. 55–66, Taejon, Korea, August 1998.
- [5] S. Grossberg, "A solution of the figure-ground problem for biological vision," *Neural Networks*, vol. 6, no. 4, pp. 463–483, 1993.
- [6] C. Harris and B. Buxton, "Evolving edge detectors with genetic programming," in *Proceedings of the 1st Annual Conference on Genetic Programming (GP '96)*, pp. 309–314, MIT Press, Cambridge, Mass, USA, July 1996.
- [7] M. Köppen and X. Liu, "Texture detection by genetic programming," in *Proceedings of the IEEE Conference on Evolutionary Computation (CEC '01)*, vol. 2, pp. 867–872, Seoul, South Korea, May 2001.
- [8] M. Köppen and B. Nickolay, "Genetic programming based texture filtering framework," in *Pattern Recognition in Soft Computing Paradigm*, N. R. Pal, Ed., vol. 2 of *FLSI Soft Computing Series*, pp. 275–304, World Scientific, Singapore, 2001.

- [9] M. Köppen, A. Zentner, and B. Nickolay, "Deriving rules from evolutionary adapted texture filters by neural networks," in *Proceedings of IEEE International Conference on Fuzzy Systems (FUZZY '99)*, vol. 2, pp. 785–790, Seoul, South Korea, 1999.
- [10] J. R. Koza, *Genetic Programming—On the Programming of Computers by Means of Natural Selection*, MIT Press, Cambridge, Mass, USA, 1992.
- [11] J. R. Koza, *Genetic Programming II: Automatic Discovery of Reusable Programs*, MIT Press, Cambridge, Mass, USA, 1994.
- [12] B. T. Lam and V. Ciesielski, "Applying genetic programming to learn spatial differences between textures using a translation invariant representation," in *Proceedings of the IEEE Conference on Evolutionary Computation (CEC '05)*, vol. 3, pp. 2202–2209, Edinburgh, UK, September 2005.
- [13] M. Köppen, A. Soria-Frisch, and T. Sy, "Using soft computing for a prototype collagen plate inspection system," in *Proceedings of the IEEE Conference on Evolutionary Computation (CEC '03)*, vol. 4, pp. 2844–2850, Canberra, Australia, December 2003.
- [14] R. Poli, "Genetic programming for feature detection and image segmentation," in *Evolutionary Computation, AISB Workshop*, T. C. Forgarty, Ed., pp. 110–125, Springer, Brighton, UK, April 1996.
- [15] D. G. Stork, R. O. Duda, and P. E. Hart, *Pattern Classification*, John Wiley & Sons, New York, NY, USA, 2nd edition, 2000.
- [16] J. Serra, *Image Analysis and Mathematical Morphology*, Academic Press, London, UK, 1982.
- [17] J. Serra, *Image Analysis and Mathematical Morphology 2: Theoretical Advances*, Academic Press, London, UK, 1988.
- [18] S. A. Stanshope and J. M. Daida, "Genetic programming for automatic target classification and recognition in synthetic aperture radar imagery," in *Proceedings of the 7th International Conference on Evolutionary Programming (EP '98)*, pp. 735–744, Springer, San Diego, Calif, USA, March 1998.
- [19] M. Sugeno, *Fuzzy Control*, Nikkan Kogyo Shimbun-sha, Tokyo, Japan, 1988.
- [20] W. A. Tackett, "Genetic programming for feature discovery and image discrimination," in *Proceedings of the 5th International Conference on Genetic Algorithms (ICGA '93)*, S. Forrest, Ed., pp. 303–309, Morgan Kaufmann, Urbana-Champaign, Ill, USA, June 1993.
- [21] R. R. Yager, "On ordered weighted averaging aggregation operators in multi-criteria decision making," *IEEE Transaction on Systems, Man and Cybernetics*, vol. 18, no. 1, pp. 183–190, 1988.

Mario Köppen: Department of Artificial Intelligence, Faculty of Computer Science and Systems Engineering, Kyushu Institute of Technology, 680-4 Kawazu, Iizuka-shi, Fukuoka 820-8502, Japan
 Email: mkoepen@pluto.ai.kyutech.ac.jp

Raul Vicente-Garcia: Department Automation Technologies, Fraunhofer-Institute for Production Systems and Design Technology, Pascalstr. 8-9, 10587 Berlin, Germany
 Email: raul.vicente@ipk.fraunhofer.de

12

A practical review on the applicability of different evolutionary algorithms to 3D feature-based image registration

Oscar Cordón, Sergio Damas,
and José Santamaría

12.1. Introduction

Image registration (IR) [9, 59] is a fundamental task in computer vision used to finding either a spatial *transformation* (e.g., rotation, translation, etc.) or a correspondence (matching of similar image entities) among two (or more) images taken under different conditions (at different times, using different sensors, from different viewpoints, or a combination of them), with the aim of overlaying such images into a common one. Over the years, IR has been applied to a broad range of situations from remote sensing to medical images or artificial vision and CAD systems, and different techniques have been independently studied resulting in a large body of research.

IR methods can be classified in two groups according to the nature of images: *voxel*-based IR methods (also called *intensity*-based), where the whole image is considered for the registration process; and, on the other side, *feature*-based methods, which consider prominent information extracted from the images, being a reduced subset of them. The latter methods take advantage of the lesser amount of information managed in order to overcome the problems found in the former when the images present some inconsistencies to deal with, for example, regardless of changes in the geometry of the images, radiometric conditions, and appearance of noise and occlusion. These features correspond to *geometric primitives* (points, lines, surfaces, etc.) which are invariant to the transformation to be considered between the input images. Moreover, the latter methods perform faster than the former ones due to the reduced amount of data they take into account, at the expense of achieving *coarse* results.

Likewise, IR is the process of finding the optimal spatial transformation (e.g., rigid, similarity, affine, etc.) achieving the best fitting/overlaying between two (or more) different images named *scene* and *model* images. They both are related with the latter transformation, measured by a *similarity metric* function. Such

transformation estimation is interpreted into an iterative optimization procedure in order to properly explore the search space. Two search approaches have been considered in the IR literature: *matching-based*, where the optimization problem is intended to look for a set of correspondences of pairs of those more similar image entities in both the scene and the model images, from which the registration transformation is derived; and the *transformation parameter-based*, where the strategy is to try to directly explore inside each range of the transformation parameters. Both strategies can be used with either a voxel-based or a feature-based approach.

Aspects such as the presence of noise in images, image discretizations, orders of magnitude in the scale of the IR transformation parameters, the magnitude of the transformation to be estimated cause difficulties for traditional local optimizers (gradient- and nongradient-based) and they become prone to be trapped in local minima. Hence, the application of several well-known evolutionary algorithms (EAs) (see Section 12.3) to the IR optimization process has introduced an outstanding interest in order to solve those problems due to their global optimization techniques nature. The first attempts to solve IR using *evolutionary computation* [5] (EC) can be found in the early eighties. Fitzpatrick et al. [25] proposed such approach based on a genetic algorithm (GA) [27] for the 2D case and applied it to angiographic images in 1984. Since then, several evolutionary approaches have been proposed to solve the IR problem.

In this chapter, we develop a practical review of the most relevant contributions (evolutionary and nonevolutionary techniques) that deal with the 3D feature-based IR problem using similarity transformations. Techniques based on both the matching and the transformation parameters approaches will be considered. Likewise, the experiments considered refer to four different realistic brain images, commonly used in IR literature for testing new IR methods, as well as to four significant test transformations to be estimated. Results are obtained always under the same conditions for all the IR methods considered, taking into account a number of different runs in order to avoid execution dependence. Because of the random nature of the EAs, it is mandatory to perform a minimum number of runs in order to guarantee the reproducibility of the experiment results. With this work, we want to highlight the performance of the EC paradigm to solve the IR problem and to encourage researchers in this field (and others) to increase the performance achieved by the present methods by contributing with new and improved EC mechanisms.

Addressing the structure of this chapter, in Section 12.2, we give some IR basics analyzing each of the principal components of a general IR method, and underlining those vulnerable aspects found by traditional IR methods. Next, we make a critical review of the existing evolutionary approaches to the IR problem in Section 12.3. Then, we develop a practical review in Section 12.4 where we make a broad benchmarking to test those IR methods which we consider most relevant in the literature together with some of our proposals in the field. Finally, in Section 12.5, conclusions and future open lines are addressed.

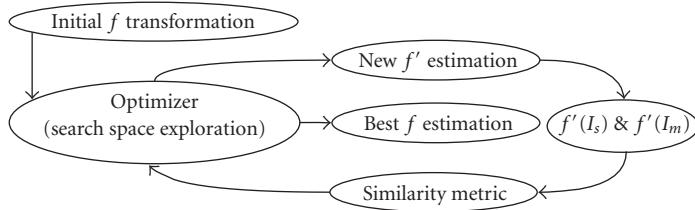


FIGURE 12.1. The IR optimization process.

12.2. The image registration problem

An extensive survey on IR methods is out of the aim of this contribution. Nevertheless, we want to introduce the key concepts related to the IR methodology. During the last decades, many different taxonomies have been established to classify the huge amount of IR methods presented so far [9, 59], considering different criteria: the image acquisition procedure, the search strategy, the type of transformation relating the images, and so forth.

Besides, there is not a universal design for a hypothetical IR method that could be applicable to all registration tasks, since various considerations on the particular application must be taken into account. However, IR methods usually require the four following components (see Figure 12.1): two input *Images* (see Section 12.2.1) named as Scene $I_s = \{\vec{p}_1, \vec{p}_2, \dots, \vec{p}_n\}$ and Model $I_m = \{\vec{p}'_1, \vec{p}'_2, \dots, \vec{p}'_m\}$, with \vec{p}_i and \vec{p}'_j being image points; a *registration transformation* f (see Section 12.2.2) being a parametric function relating the two images; a *similarity metric function* F (see Section 12.2.3) in order to measure a qualitative value of closeness or degree of fitting between the transformed scene image, denoted by $f'(I_s)$, and the model image; and an *optimizer* which looks for the optimal transformation f inside the defined solution search space (see Section 12.2.4). In the subsequent subsections, we will deeply analyze each of these four IR components.

Hence, the key idea of the IR process is focused on determining the unknown parametric *transformation*, that relates both images, by placing them in a common coordinate system bringing the points as close as possible. Because of the uncertainty underlying such transformation, the IR task arises as a *nonlinear problem* that cannot be solved by a direct method (e.g., resolution of a simple system of linear equations). It should be solved by means of an iterative procedure searching for the *optimal estimation* of f , following a specific search space optimization scheme aiming at minimizing the error of a given *similarity metric* of resemblance. Classical local optimizers can be used for this task although their main drawback (see Section 12.2.4) is that they usually get trapped in a local minima solution. The main reasons for such behavior are related to both the nature of the problem to be tackled and the greedy/local search features of these methods. Hence, the interest on the application of EAs to the IR optimization process has increased in the last decade due to their global optimization nature.

12.2.1. Nature of images

According to the nature of images, IR methods can be classified as *voxel*-based (or *intensity*-based) and *feature*-based [59]. While the former directly operate with the whole raw images, the latter approaches introduce a previous step: before the application of the registration process, a reduced subset of the most relevant features are extracted from the images. Since voxel-based methods can deal with a major amount of image information, they are often considered as *fine-tuning* registration processes, while feature-based methods typically achieve a *coarser* approximation due to the reduced data they take into account.

One important drawback of voxel-based approaches relies on the commonly used rectangular window for the correspondence estimation. If the images are deformed by complex transformations, this type of window will not be able to cover the same parts of the transformed scene and model images. Moreover, if the window contains a smooth image region without any prominent detail, it will probably be incorrectly matched to other smooth image region in the model image. Nevertheless, the principal disadvantage of voxel-based methods comes from situations where there are changes in illumination during the acquisition of the scene and the model images. In that case, the similarity metric offers unreliable measurements and induces the optimization process to be trapped in local minima.

With the intention of avoiding many of the drawbacks related to voxel-based methods, the second IR approach is based on the extraction of prominent geometric primitives (*features*) from the images. The proper comparison of feature sets will be possible using a reliable feature detector that confronts the accurate extraction of invariant features, that is, regardless of changes in the geometry of the images, radiometric conditions, and appearance of noise. There are many different features that can be considered, for example, *region features*, *line features*, and *point features*, among which corners are widely used due to their invariance to the image geometry.

For the final practical review of the most relevant IR methods (Section 12.4), we have decided to choose the feature-based approach, where the features considered are prominent image points extracted from magnetic resonance images of brains using local curvature information (see Section 12.4.1).

12.2.2. Transformations

We can classify IR methods according to the registration transformation model used to relate both the scene and the model images. The first category of transformation models includes *linear transformations*, which preserves the operations of vector addition and scalar multiplication, being a combination of translation, rotation, global scaling, and shear components. Among the most common linear transformations are rigid, similarity, affine, projective, and curved.

Linear transformations are global in nature, thus not being able to model local deformations. The second category of transformation models includes “*elastic*” or

“*nonrigid*” transformations. These transformations allow local warping of image features, thus providing support for local deformations.

The kind of transformation model considered will depend on the particular application addressed and the nature of images it will take into account. The registration transformation f that we will use in the latter practical benchmarking (Section 12.4) is a 3D similarity one composed of a translation, a rotation, and a uniform scaling, which has been considered to register aerial and satellite images, bony structures in medical images, and brain multimodal images [29], among other applications.

12.2.3. Similarity metric

One of the most important components of any IR method is the similarity metric. This is considered as a function F that measures the goodness of a given registration solution, that is, of a registration transformation f . The final performance of any IR method will depend on its accurate estimation.

Each solution is evaluated by F applying such transformation f to one of the two images, usually to the scene image ($f(I_s)$). Next, the degree of closeness or fitting between the transformed scene and the model images, $\Psi(\cdot)$ must be determined,

$$F(I_s, I_m, f) = \Psi(f(I_s), I_m). \quad (12.1)$$

There are many approaches trying to estimate such function $\Psi(\cdot)$ depending on the dimensionality (2D or 3D) and the nature of the considered images, for example,

- (a) voxel-based approach: sum of squared differences [7], normalized cross-correlation (i.e., correlation coefficient [52] or phase correlation [20]), and mutual information [54];
- (b) feature-based approach: feature values-based metrics (i.e., registration based on the curvature) and distance between corresponding geometric primitives [2, 4, 11, 47].

Like in the previous IR components, the F function is affected by both the discretization of images and the presence of noise, causing worse estimations and favoring the IR method to get trapped in local minima.

Notice that the huge amount of data (images) required in some applications makes the problem-solving very complex and the IR procedure very time-consuming. Therefore, in order to speed up the similarity metric computation, most of the IR contributions use any *spatial indexing* data structure in order to improve the efficiency of the considered optimization method, each time the closest point assignment computation between the transformed scene and model images must be done. Likewise, this data structure is computed only once at the beginning of the IR method. Two main variants of spatial indexes can be found in the 3D IR literature as follows.

(i) *Kd-tree*, which consists in a generalization of bisection in one dimension to k -dimensions in order to build, in the 3D case ($k = 3$), a binary tree that successively cuts the whole space into two rectangular parallelepipeds such that there is an approximately equal number of points on each side of the cutting plane, for the xy , xz , and yz planes. We find the first proposal of applying *kd-trees* to the IR problem in [58].

(ii) *Distance map*, a volume (typically a 3D grid) containing a surface, where each voxel (a given cell on the 3D grid) within the volume links to the closest point on that surface. Such a data structure has been widely used in computer vision and recently applied to solve IR with GAs [49]. Yamany et al. [56] considered a particular distance map, named *grid closest point* (GCP), which consists of two cubes splitting the 3D space. The first one divides it into a set of $L \times W \times H$ cells and covers the two images. The second one only encapsulates the model image within a rectangular volume of double resolution ($2L \times 2W \times 2H$ cells). The goal of this second grid is to reduce the discretization error of the first one, in order to achieve more accurate outcomes in the final stages of the IR process.

12.2.4. Search space strategies

The derivation of the registration transformation f that relates two given images has received a great deal of attention during the past. A good review on the topic can be found in [9, 59] according to transformation estimation.

The IR process performs an iterative exploration to obtain that optimal transformation f (previously introduced in Figure 12.1). So, the closer f to the unknown global optimum, the better the fitting (measured by the similarity metric F) between scene and model. The optimization process considered to obtain those solutions can be deterministic or stochastic (either a global or a local one).

Although the final registration problem solution consists of the right values for the parameters which determine f , we can distinguish, from the IR literature, two different strategies to solve the problem, each of them working in a different solution space: (i) the first searches in the *matching* space to obtain a set of correspondences of pairs of the most similar image entities in both the scene and the model images, from which the registration transformation is derived; and (ii) the second directly makes a search in the space of the f parameters guided by the F function, called *transformation parameters* space. We will analyze both strategies more deeply as follows.

12.2.4.1. Matching-based search space

This search space exploration strategy needs to compute the two following steps at once: first, a set of matchings (correspondences) with those more similar pairs of regions of pixels (voxel-based) or geometric primitives (feature-based) in both the scene and the model images must be established; next, the transformation f is retrieved by numerical methods considering such matching. *Least squares* (LS) estimators are the most commonly used numerical methods [3, 23, 34], due to their

special and interesting properties, for example, they only require means, variances, and covariances to be finite [41].

In the classical theory of estimation, the notion of outliers is vague. They can be interpreted as erroneous (noisy) observations which are well separated from the bulk of the data, thus demanding special attention. Besides, we assume that outliers will not provide any outstanding information about f parameters. On the contrary, they can damage their correct estimation.

LS estimators assume that the observation of errors must be normally distributed to perform correctly. In the related literature, we can find some works proposing extensions of the LS estimator based on the analysis of residuals of the *L2 norm* (least squares) to identify erroneous observations [21, 26]. Since outliers have an unknown distribution of observations, this kind of estimators cannot guarantee inferring the true transformation, thus a robust estimator may be better suited. For instance, the Danish method [36] could be considered, being a reweighting scheme based on a predefined function that disallow errors by avoiding their corresponding weights, or the well-known *M-estimators* technique [35], widely used due to their robustness.

Therefore, the complexity of the matching step and of the following registration transformation estimation depends on the method being considered. Likewise, an iterative process may be followed either for the estimation of the matching, or the registration, or both, until convergence within a tolerance threshold of the concerned similarity metric. This is the case of the *iterative closest point (ICP)* algorithm [8], well-known in CAD systems and originally proposed to recover the 3D transformation of pairs of range images. Next, we will briefly describe the structure of this local optimizer in order to get a better understanding of the strategy. The method proceeds as follows.

(i) A point set P with N_p points \vec{p}_i (cloud of points) from the data shape (scene) and the model X —with N_x supporting geometric primitives: points, lines, or triangles—is given. The original paper dealt with 3D rigid transformations stored in the solution vector

$$\vec{q} = [q_1, q_2, q_3, q_4, t_1, t_2, t_3]^t, \quad (12.2)$$

where the first four parameters corresponded to the four components of a quaternion determining the 3D rotation, and the last three parameters stored the translation vector.

(ii) The procedure is initialized by setting $P_0 = P$, the initial registration transformation to $\vec{q}_0 = [1, 0, 0, 0, 0, 0, 0]^t$, and $k = 0$. The next four steps are applied until convergence within a tolerance $\tau > 0$.

- (1) Compute the matching between the data (scene) and model points by the closest assignment rule: $Y_k = C(P_k, X)$.
- (2) Estimate the registration by least squares: $f_k = \varrho(P_0, Y_k)$.
- (3) Apply the registration transformation to the scene image: $P_{k+1} = f_k(P_0)$.
- (4) Terminate iteration if the change in *mean square error* (MSE) falls below τ . Otherwise, $k = k + 1$.

The original ICP proposal has two main drawbacks: (i) one of the two images (typically the scene one) should be contained in the other, for example, in feature-based IR problems, the geometric primitives of one image should be a subset of those in its counterpart image; and as it has been previously said, (ii) it cannot handle non-normally distributed observations. Since it was introduced, many contributions have been proposed extending and partially solving the latter problems of the original proposal [24, 39, 58]. On the other hand, other proposals of matching-based IR approaches are to be found in [15, 18].

Notice how the operation of these matching-based IR methods is not guided by the similarity metric but by the computed matching. In this strategy, the function F (MSE) only plays the role of the stopping criterion. Moreover, the transformation estimator (numerical method) is dependent on good choices performed in the matching step. So, the better the choice of the matchings done, the more precise the transformation f and, consequently, the more accurate the value of the similarity metric will be for a proper convergence.

For the later practical benchmarking, among other methods, we have considered our point-matching contribution that makes use of the scatter search [38] (SS) EA. Unlike the ICP-based proposals, the SS-based approach is not so dependent on the initial orientation of the scene and model images.

12.2.4.2. Transformation parameters-based search space

Opposite to the previous approach, the second one involves directly searching for the solution in the space of parameters of the transformation f . To do so, in this strategy, each solution to the IR problem is encoded as a vector composed of each one of the values for the parameters of f .

In this way, the IR method works by generating possible vectors of parameter values, that is, possible registration transformations. Likewise, the second principal major difference with ICP-based strategies is that the search space exploration is guided by the similarity metric F . Each solution vector is evaluated by such metric thus clearly stating the IR problem as the parameter optimization procedure of finding the best values defining f that minimize F .

It is interesting to highlight the fact that the orders of magnitude in the scale of f parameters are crucial for IR methods dealing with this search space strategy. Unit changes in angle have much greater impact on an image than unit changes in translation. Indeed, when applying a rotation, the further a given point on the image from its center of mass (origin of rotation), the greater the displacement. Meanwhile, in case of translations, the distance between the transformed scene and the model images is kept constant. This difference in scale appears as elongated valleys in the parameter search space, causing difficulties for the traditional gradient-based local optimizers [8, 30]. Thus, if the considered IR method is not robust tackling these scenarios, the theoretical convergence of the procedure is not guaranteed, and in most cases, it will be trapped in local minima.

Together with the commonly used local (gradient- and nongradient-based) optimizers [42], EAs (see Section 12.3) are the most used optimization procedure

for IR when working with this search space strategy, as can be seen from the large number of contributions made [10, 13, 15, 17, 30, 45, 49, 51, 56, 57]. Unlike IR methods based on local optimizers, the main advantage of using EA-based IR methods is that they do not require a solution near to the optimum one to achieve high-quality registration results.

12.3. Evolutionary computation and image registration

As we will briefly depict in this section, the application of EAs to the IR optimization process has caused an outstanding interest in the last few years. Thanks to the nature of their global optimization techniques, EAs aim at solving the problems described in the latter sections, not satisfactorily tackled by traditional IR methods.

This section is devoted to make an analytical review of the different EC-based IR approaches proposed in the literature. Some of them will be considered for our later experimental study based on 3D feature-based IR applied to brain images (Section 12.4).

The first attempts to solve IR using global optimization techniques can be found in the early eighties. The size of data, as well as the number of parameters that are looked for, prevents from an exhaustive search for the solutions. An approach based on a GA was proposed in 1984 for the 2D case and applied to angiographic images [25]. Since this initial contribution, different authors solved the problem but we can still find important limitations in their approaches.

The first of them is the use of a binary coding to solve an inherent real-coding problem. This leads us to the situation of having to balance the magnitude of the transformation being considered and the precision of the returned solution, for a given number of bits in the encoding. Such situation can only be allowable in environments where precision is not a critical requirement or where we know the range of change in the parameters of the transformation. Moreover, if we try to get a more accurate solution increasing the number of bits of the encoding, then the required time of the algorithm to converge will rise as well.

For instance, in 1989, Mandava et al. [44] used a 64-bit structure to represent a possible solution when trying to find the eight parameters of a bilinear transformation. Tsang [53] used 48-bit chromosomes to encode three test points as a base for the estimation of the 2D affine registration function. In the more recent proposals by Yamany et al. [56] and Chalermwat et al. [10], the same binary coding is found when dealing with 3D and 2D rigid transformations, respectively. Yamany et al. enforced a range of $\pm 31^\circ$ over the angles of rotation and ± 127 units in displacement by defining a 42-bit chromosome with eight bits for each translation parameter and six bits for each rotation angle. Meanwhile, Chalermwat et al. used twelve bits for the coding of the 2D rotation parameter to get a search scope of $\pm 20.48^\circ$, therefore allowing the use of a precision factor for the discretization of the continuous rotation angle interval, while other ten bits stored each of the two translation parameters (± 512 pixels).

Apart from the use of the basic binary coding, the kind of GA considered is usually based on the original proposal by Holland [33]. In this way, a selection strategy based on fitness-proportionate selection probability assignment and the stochastic sampling with replacement, as well as the classical one-point crossover and simple bit flipping mutation, are used. On the one hand, it is well known that such a selection strategy causes a strong selective pressure, thus having a high risk of premature convergence of the algorithm. On the other hand, it has also been demonstrated that it is difficult for the single-point crossover to create useful descendants as it is excessively disruptive with respect to the building blocks [27].

Another major drawback of many contributions is that they only handle images related to a rigid transformation [13, 30, 51]. The use of such transformations can be compatible with a limited number of applications, but this is not the case in many real situations where at least a uniform scaling is desirable. Different proposals have been made in the 2D case [44, 45, 53] to consider either bilinear or affine transformations.

In addition, we would like to analyze in depth three recent and very interesting contributions.

In their proposal [49], Rouet et al. face 3D MR-CT registration by means of a three-step algorithm. First, global rigid transformation parameters are determined by a GA working directly on real numbers. Second, they use another point-matching GA trying to determine a global trilinear transformation. Last, they introduce a post-analysis of the output population of the previous step in order to achieve a fine tuning of the solution using a local optimization process. This proposal provides one of the most overall approaches to the IR problem using GAs we have found in the literature. Nevertheless, we still identify different weak points.

- (i) Different studies have shown that a balance between population diversity and convergence to the solution is needed in order to get a good behavior of any GA (to avoid being trapped in local minima). Although Rouet et al. used the principle of Latin squares¹ to control the distribution of the population over the search space, there are other approaches that can perform this task better. Niching techniques [19, 28, 27, 43], or even the principles of the CHC² algorithm [22] are well-known schemes for the EC community and seem to be more suitable approaches.
- (ii) Moreover, the success of the second step of the algorithm depends on a precise definition of the curvature class of each point.
- (iii) Finally, the use of simple operators (like uniform crossover) in a real-coded GA is not the best option even if we want to improve the efficiency of our algorithm [32].

He's and Narayana's proposal [30] denoted by GA_{He} in the experimental study developed in this contribution, it is a slight improvement of Yamany et al.'s [56] approach, proposing a real-coding scheme that makes use of an arithmetic

¹This procedure is not known in the EC community and we only recognize it as a diversity induction mechanism due to the authors' explanation.

²CHC stands for Cross generational elitist selection, Heterogeneous recombination, and Cataclysmic mutation.

crossover and a uniform mutation operators within an elitist generational model that considers a restart mechanism. GA_{He} deals with rigid transformations following a two-step technique making a first coarse parameter estimation using a real-coded GA, and then refining such results with the dividing rectangle method to perform a local search. In the coarse resolution, the ranges of the parameters were set to: ± 20 voxels along x - and y -directions and ± 40 voxels along z direction, and rotation of $\pm 10^\circ$ around x - and y -axes, and $\pm 20^\circ$ around z -axis. However, the setting of the parameters range as well as the rigid transformation between both images may be a weak point when trying to apply this method to some real-world environments.

Chow et al.'s GA-based proposal [12] denoted by GA_{Chow} , has the same generational and proportionate-fitness models for population reproduction than the previous one, and introduces a crossover operator that randomly selects the number of genes to be swapped. The value to be accumulated for a mutated gene is generated randomly within a constant range for the rotation genes and dynamically computed for the translation ones according to the fitness value of the chromosome. It makes also use of GAs with more suitable components to the current EC framework such as a real-coding scheme and a sophisticated restart mechanism ("dynamic boundary"). In spite of these improvements, there are some drawbacks in terms of accuracy, as the authors work with a smaller randomly selected data set from scene images with a huge amount of data. Besides, although the algorithm aims at getting a quick registration estimation with the latter procedure, the efficiency could be reduced since it needs to perform a sort operation for each evaluation of the fitness function. As in many of the mentioned proposals, it also has the limitation of only considering a rigid transformation (translation and rotation). Finally, the restarting scheme assumes that, prior to its application, the population will fall in a search space zone located into or near to the global optimum.

On the other hand, several approaches based on the use of advanced EAs able to solve the said problems have been introduced by our research group and are briefly described as follows.

SS [38] is based on a *systematic* recombination between solutions from a *Reference Set*, instead of a randomized one like the one usually carried out in EAs, and on a local optimization of the solutions. We have adopted this approach to propose our SS -based registration method (denoted by SS_m) [18] in a combinatorial optimization fashion that makes use of problem dependent (context) information by taking into account the curvature information extracted from the tackled images. We proposed new designs for three of the five SS components, the generator of diverse solutions, and both the improvement and the combination methods, to develop a method with improved performance working on the matching search space.

As said, the main novelty of our SS_m contribution is that the heuristic values of the features of the image isosurfaces (the curvature information that will be seen in Section 12.4.1) are used to guide the matching. So, we defined a function $m_{\text{error}}(\cdot)$ evaluating the goodness of the matching stored in a given solution coded

as a permutation, denoted by π , by using the said curvature values. In [18], we chose the following:

$$m_{\text{error}}(\pi) = \Delta k_1 + \Delta k_2, \quad \text{where } \Delta k_j = \sum_{i=1}^r (k_j^i - k_j^{\pi_i})^2, \quad j = \{1, 2\}, \quad (12.3)$$

Δk_1 and Δk_2 measure the error associated to the curvature matching of scene and model points with different values for the first and second principal curvatures, respectively.

Meanwhile, the objective function of the SS_m method will include both information regarding the usual IR measure $g(\pi)$ (mean square error of point matching) and the previous criterion as follows:

$$\min F(\pi) = w_1 \cdot g(\pi) + w_2 \cdot m_{\text{error}}(\pi), \quad (12.4)$$

where w_1, w_2 are weighting coefficients defining the relative importance of each.

As in *CHC EA* [16], denoted by CHC_{bin} , we used the sophisticated CHC EA for the registration of *magnetic resonance images (MRIs)*. To do so, we made use of binary-coded solutions and the HUX crossover. The second of our proposals in [16], denoted by CHC_{real} , was based on the afore-mentioned approach but extending it to work in a real-coded fashion as well as using different operators as the BLX- α crossover. In that contribution, f was a similarity transformation. Therefore, the objective/fitness function of the considered transformation parameters-based IR algorithms was slightly extended to take into account the scaling parameter of such f , since most of these IR methods only dealt originally with rigid transformations,

$$F'(f', I_s, I_m) = \omega_1 \cdot \left(\frac{1}{1 + \sum_{i=1}^N \|(sR\vec{p}_i + \vec{t}) - \vec{p}_j^*\|^2} \right) + \omega_2 \cdot \left(\frac{1}{1 + |\rho_c^s - \rho^m|} \right), \quad (12.5)$$

where I_s and I_m are the scene and model images; f' is the transformation encoded in the evaluated solution; \vec{p}_i is the i th 3D point from the scene and \vec{p}_j^* is its corresponding nearest point in the model obtained with the spatial index data structure; ω_1 and ω_2 ($\omega_1 + \omega_2 = 1$) weight the importance of each function term; ρ_c^s is the radius of the sphere wrapping up the scene image transformed with the current f ; and ρ^m is the radius of the sphere wrapping up the model image. As the first term of F' reveals, the error modelled corresponds to the MSE one. Note that F' maximizes to 1.0 for a rarely perfect fit.

The *SS in the transformation parameters-based search space* [17]. We introduced a particular design of the SS EA to register MRIs, denoted by SS_p , for example, using the BLX- α combination method, the Solis and Wets' local optimizer for the improvement method, and a frequency-based solution generator for the generator of diverse solutions.

12.4. Benchmarking of 3D feature-based IR methods

This section is devoted to developing a practical review of the most relevant (evolutionary and nonevolutionary) solutions to the 3D feature-based IR problem using the similarity transformation. Both matching-based and transformation parameters-based search strategies will be considered with the main goal of comparing the ICP-based contributions with the remaining ones dealing with complex IR scenarios.

12.4.1. 3D images considered

As said, our results correspond to a number of registration problems with four different 3D test brain images. These images have been obtained from the BrainWeb database at McGill University [14, 37]. The purpose of this simulator is to provide researchers with ground truth data for image analysis techniques and algorithms. BrainWeb has been widely used by the IR research community [31, 48, 55].

To establish a more realistic scenario, every medical image under consideration corresponds to one MRI. Moreover, we have added different levels of noise to three of the four images used to model noisy conditions related to the images acquired by some devices. Likewise, we cannot neglect one of the most important goals of IR supporting critical decisions concerning the evolution of a patient's lesion. To do so, two of our images will include a multiple sclerosis lesion. The influence of these two factors (the noise intensity and the presence or absence of lesion) will allow us to design a set of experiments with different complexity levels.

A preprocessing step has been applied to all these 3D images in order to obtain problem dependent information to guide the IR process as well as to reduce the huge amount of data stored in the initial instances of the images. Therefore, we extract the isosurface and select crest-line points [46] with relevant curvature information from both the scene and model images.

The first image (I_1 , shown at the top left of Figure 12.2) corresponds to an MRI of a healthy person obtained with an ideal scanner, that is, no lesion is present and it is a noise-free scenario. After the isosurface extraction to identify the brain and the crest-line points study to choose those features with relevant curvature information, 583 points have been selected. Image I_2 (second row of Figure 12.2) corresponds to a low level of noise scenario (1% of Gaussian noise) of a healthy person. After using the isosurface extraction, 393 crest-line points have been chosen. The third image (I_3 , see first cell at the third row of Figure 12.2) includes a multiple sclerosis lesion (located within the circle) and the same level of noise of I_2 . After the isosurface extraction to identify the brain and the crest-line points study to choose those features with relevant curvature information, 348 points have been selected. Finally, image I_4 (the last one at the bottom left of Figure 12.2) corresponds to a multiple sclerosis patient where the MRI has been acquired using a poor device (5% of Gaussian noise is introduced). Blurring can be easily observed in the extracted isosurface. Finally, 284 points with relevant curvature have been identified.

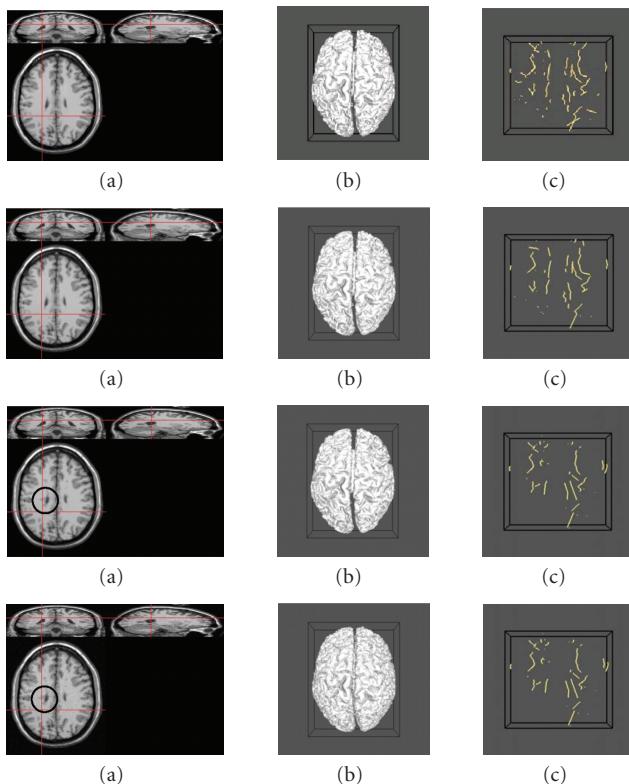


FIGURE 12.2. From top to bottom: the images I_1 , I_2 , I_3 , and I_4 . For all of them, (a) original MRI with three views (transverse, sagittal, and coronal). Different organs (skull, brain, eyes, etc.) can be clearly identified. (b) Isosurface corresponding to the brain from the MRI. (c) Crest line points with relevant curvature information.

12.4.2. IR problems considered

Our results correspond to a number of IR problem instances for the different 3D images presented above which have suffered the same four global similarity transformations (denoted by T_1 , T_2 , T_3 , and T_4 in Table 12.1), to be estimated by the different 3D IR algorithms applied (see Section 12.4.3). These are ground truth transformations and they will allow us to quantify the accuracy of the IR solution returned by every algorithm. Hence, we will know in advance the optimal (i.e., the true) registration transformation relating every scene and model input, thus being able to compute the fitness function value (or the similarity metric) associated to the optimal problem solution (see Section 12.4.5).

As mentioned in Section 12.2.2, similarity transformations involve rotation, translation, and uniform scaling. They can be represented by eight parameters: one for the rotation magnitude λ , three for the rotation axis (axis_x , axis_y , axis_z), three for the translation vector (t_x , t_y , t_z), and one more for the uniform scaling

TABLE 12.1. Global similarity transformations applied to every 3D image.

	T_1	T_2	T_3	T_4
λ	115.0	168.0	235.0	276.9
axis_x	-0.863868	0.676716	-0.303046	-0.872872
axis_y	0.259161	-0.290021	-0.808122	0.436436
axis_z	0.431934	0.676716	0.505076	-0.218218
t_x	-26.0	6.0	16.0	-12.0
t_y	15.5	5.5	-5.5	5.5
t_z	-4.6	-4.6	-4.6	-24.6
s	1.0	0.8	1.0	1.2

TABLE 12.2. From top to bottom: increasing complexity ranking of the IR problem scenarios considered.

IR problem	Scene image		Model image	
	Lesion	Noise	Lesion	Noise
I_1 versus $T_i(I_2)$	No	No	No	1%
I_1 versus $T_i(I_3)$	No	No	Yes	1%
I_1 versus $T_i(I_4)$	No	No	Yes	5%
I_2 versus $T_i(I_4)$	No	1%	Yes	5%

factor, s . In order to achieve a good solution, every algorithm must estimate these eight parameters accurately. Values in Table 12.1 have been intentionally selected as complex transformations to be estimated. Both rotation and translation vectors represent a strong change in the object location. In fact, the lowest rotation angle is 115° . Meanwhile, translation values are also high. Likewise, the scaling factor ranges from 0.8 (in the second transformation) to 1.2 (in the fourth one). This way, complex IR problem instances are likely to be generated.

Moreover, in order to deal with a set of problem instances with different complexity levels (see Table 12.2), we will consider the following scenarios (from lower to higher difficulty): I_1 versus $T_i(I_2)$, I_1 versus $T_i(I_3)$, I_1 versus $T_i(I_4)$, and I_2 versus $T_i(I_4)$. Therefore, every algorithm will face sixteen different IR problem instances, resulting from the combination of the four scenarios and these four different transformations T_i .

12.4.3. IR methods considered

This subsection is devoted to describe the different IR methods to be compared in the subsequent practical benchmarking. Methods belonging to the matching and the transformation parameters search space strategies, as well as having either an

evolutionary or nonevolutionary nature will be considered (see Section 12.4.1 for all the descriptions of the different EA-based IR approaches).

Concerning the considered matching-based IR methods, *I-ICP* and *ICP + SA* are two of the nonevolutionary techniques found in the specialized literature. The former IR method considers the rejection of false matchings as robust mechanism for the point-matching step and inspired on the *colinearity constraint*. *I-ICP* was proposed by Liu [39] for addressing the registration of *range images*. On the other hand, the latter was proposed by Luck et al. [40] for the *pairwise* IR of range images as an hybrid approach that combines an ICP-based algorithm with a Simulated Annealing [1] procedure within an iterative process. Moreover, we have considered our proposal inspired on the SS EA (SS_m) as the third matching-based IR method.

With respect to the transformation parameters-based IR methods, three well-known EA-based IR methods as well as three proposals developed by the authors are used. Yamany et al. [56], denoted by GA_{Yamany} , proposed a binary-coded GA with a generational model for population reproduction and a proportionate-fitness selection together with a multipoint crossover operator. Two other transformation parameters-based IR methods considered are those from He and Narayana [30] (GA_{He}) and Chow et al. [12] (GA_{Chow}). Our three own proposals following the transformation parameters-based approach SS_p , CHC_{real} and CHC_{bin} were described in Section 12.3.

In our experimentation similarity transformations will be considered because they are the most appropriate for the current medical application. Since some of the EA-based IR methods were constrained to use rigid transformations, there is a need to extend them to make them able to deal with the latter kind of registration transformations. Due to this reason, all of the evolutionary transformation parameters-based IR methods and the *ICP + SA* annealing stage will make use of the same objective function, that shown in (12.5). Thanks to this, we will be able to develop a fair comparison respect to our previous proposals³.

All the previous IR methods have been implemented with the C++ programming language and compiled with GNU/gcc without code optimization options.

12.4.4. Parameter settings

Before performing the final experimentation, we have just made a preliminary study on the most suitable parameter values for the different IR algorithms to be considered. Both the preliminary tests and the subsequent experimentation have been made on a platform with an Intel Pentium IV 2.6 GHz processor.

For the I-ICP algorithm, we set a maximum of 40 iterations to ensure its success under some needed favourable initial conditions. For example, an initial registration transformation close to the ground truth is needed by the algorithm to achieve good results. Unfortunately, such information from which an optimal

³Note that GA_{Chow} slightly modifies the first term of (12.5) to consider the median square error.

starting point could be inferred is not usually available, thereby we have chosen an arbitrary rotation, a translation given from the subtraction of both scene and model centroids, and a uniform scaling factor estimated as in [34]. Furthermore, the parameter associated to I-ICP has been set to the same value the author used in [39].

For the ICP + SA algorithm, a maximum of 40 iterations for the wrapped I-ICP algorithm has also been fixed. The annealing process considers a maximum of 50 trial solutions for each one of the 20 annealing iterations, with an initial temperature value estimated with $T_0 = [\mu / -\ln(\phi)]C(S_0)$, where $C(S_0)$ is the cost of the given solution generated by the previous run of I-ICP, and both the μ and the ϕ factors take value 0.3. Each ICP + SA two-step iteration involves subsequently applying the I-ICP algorithm, optimizing the previous solution generated by the annealing method (if it achieves a best solution reducing the error function), and starting again a new iteration. We have only considered one iteration for the ICP + SA procedure.

Our SS-based point matching proposal, SS_m , deals with an initial set P comprised 80 diverse solutions, and a *RefSet* composed of $b = b_1 + b_2 = 10$ solutions, with $b_1 = 7$ in the *Quality subset* and $b_2 = 3$ in the *Diversity subset*. The local search algorithm putting into effect the *Improvement Method* is run for a maximum of 80 iterations.

In the Dynamic GA (GA_{Chow}), we set the size of the initial population to 100 individuals and maintained the remaining specific parameters with their original values [12].

The size of the initial population is 100 individuals, also for GA_{Yamany} , CHC_{bin} , CHC_{real} , and GA_{He} . As regards the number of bits associated to each gene in both the GA_{Yamany} and the CHC_{bin} algorithms, we have preferred to consider large individuals as we want precise solutions even if we use a wide range for each transformation parameter. This leads us to define the binary chromosome (solution to the IR problem) as an 80-bit structure: ten bits for each of the eight registration transformation parameters. On the other hand, the value of the parameter α in the BLX- α crossover, employed in both CHC_{real} and GA_{He} , is consequently set to 0.5, in order to give them a more suitable balance between the exploitative and the explorative nature for their search space exploration. Thereafter, crossover and mutation probabilities are set to $P_c = 0.6$ and $P_m = 0.1$ for both GA_{Yamany} and GA_{He} .

The parameter settings for SS_p follows. The diverse set P is initialized with 30 solutions ($Psize = 30$) and the *RefSet* is composed of the $b = 8$ best ones of them, according to quality criterion. BLX- α is applied with $\alpha = 0.3$, while the improvement method is selectively applied during 80 evaluations.

Unlike the I-ICP and the ICP + SA methods, we have established a maximum CPU time of 20 seconds for each run of the remaining algorithms. Furthermore, for each one of the EC-based registration methods as well as for the ICP + SA one, we have performed a total of 15 runs (with different random seeds) for each of the sixteen problem instances, in order to avoid the usual random bias of probabilistic algorithms.

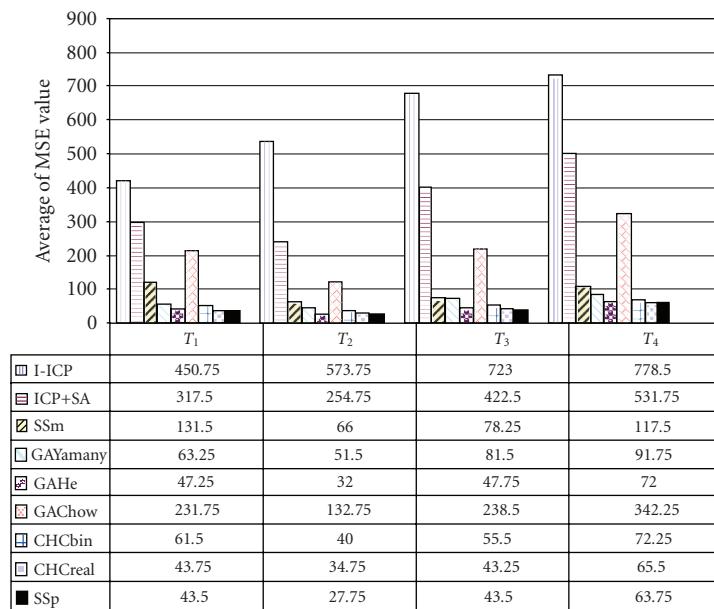


FIGURE 12.3. Bar graph measuring the robustness degree of the IR methods considered.

12.4.5. Analysis of results

Notice that all statistics in this section are based on a typical error measure in the IR field, the *mean square error* (MSE), given by

$$\text{MSE} = \frac{\sum_{i=1}^N \|f(\vec{p}_i) - \vec{p}_j\|^2}{N}, \quad (12.6)$$

where f is the estimated similarity transformation, \vec{p}_i are the scene image points, and \vec{p}_j are the model image points matching the scene points (the closest to the former).

With the aim of showing the robustness of the considered IR methods, the performance obtained by each of them is graphically depicted in Figure 12.3. Such bar graph corresponds to the average value of the final $4 \cdot 15 = 60$ MSE values achieved in every run for each of the four IR scenarios tackled, fixing one of the four test transformations considered.

In Figure 12.3, we can see how the IR methods based on the matching approach (I-ICP, ICP + SA, and SS_m) obtain worse registration estimations for each of the four transformations than those achieved by the methods based on the transformation parameters approach, except those estimated by GA_{Chow} . Such behavior demonstrates the flaws of the matching-based methods and their poor performance when they must deal with not handled initial conditions present in the images. In spite of this, we want to highlight the success of our SS_m method. It

TABLE 12.3. MSE corresponding to the four transformations in Table 12.1 applied to the I_1 versus $T_i(I_2)$, I_1 versus $T_i(I_3)$, I_1 versus $T_i(I_4)$, and I_2 versus $T_i(I_4)$ IR problems considered. The symbol m stands for the minimum, M for the maximum, μ for the mean, and σ for the standard deviation values.

T_1	m	M	μ	σ	T_2	m	M	μ	σ
GA _{He}	33	43	38	3	GA _{He}	21	50	27	9
CHC _{real}	32	32	32	0	CHC _{real}	21	49	30	13
SS _p	32	32	32	0	SS _p	21	21	21	0
I_1 versus $T_i(I_2)$									
T_3	m	M	μ	σ	T_4	m	M	μ	σ
GA _{He}	32	42	35	2	GA _{He}	47	115	55	17
CHC _{real}	32	32	32	0	CHC _{real}	47	47	47	0
SS _p	32	32	32	0	SS _p	47	47	47	0
I_1 versus $T_i(I_3)$									
T_3	m	M	μ	σ	T_4	m	M	μ	σ
GA _{He}	67	83	71	4	GA _{He}	42	48	44	2
CHC _{real}	64	64	64	0	CHC _{real}	41	85	44	11
SS _p	64	65	64	0	SS _p	41	42	41	0
I_1 versus $T_i(I_4)$									
T_3	m	M	μ	σ	T_4	m	M	μ	σ
GA _{He}	49	67	54	5	GA _{He}	71	84	75	4
CHC _{real}	48	48	48	0	CHC _{real}	70	70	70	0
SS _p	48	48	48	0	SS _p	69	70	70	0
I_2 versus $T_i(I_4)$									
T_3	m	M	μ	σ	T_4	m	M	μ	σ
GA _{He}	29	33	30	1	GA _{He}	18	49	21	7
CHC _{real}	29	75	32	12	CHC _{real}	18	48	30	0
SS _p	29	29	29	0	SS _p	18	18	18	0

offers the best results with respect to I-ICP, ICP + SA, and GA_{Chow} in every case, as well as to GA_{Yamany} considering the T_3 transformation.

It could be suspicious that, although GA_{Chow} makes use of a real-coded representation and it is supposed to be a more recent and suitable EA, previous IR methods as GA_{Yamany}, GA_{He}, and our CHC_{bin} and CHC_{real} obtain better global results with respect to the former. The reason is the initial hypothesis of GA_{Chow} when the process converges to local optima, the global optimum is in the restricted region of the reestablished search space defined after the application of the proposed *dynamic boundary* mechanism. Since we are dealing with large transformations, the previous hypothesis is not guaranteed.

From Figure 12.3, three best performing algorithms can be addressed, GA_{He} our CHC_{real}, and SS_p proposals. Table 12.3 shows the statistics of these three IR methods, highlighting the best results achieved by the most robust IR method among those analyzed, SS_p.

12.5. Concluding remarks

This chapter addresses the IR problem from the point of view of the applicability of evolutionary schemes. Firstly, the definition of the IR problem as well as a nonextensive description of each of its components, together with the main drawbacks present in the traditional IR methods, have been introduced. Next, a review of the most relevant EC-based IR algorithms has been done to establish the state-of-the-art in this field, underlining the weak points of current and previous proposals to encourage researchers to increase the quality of results achieved by the present IR methods.

Finally, this work ends with a practical benchmarking of the most relevant IR methods (in our modest opinion) facing the 3D feature-based IR problem by developing a broad experimentation using several MRIs of human brains. The results obtained highlight the better performance offered by the evolutionary IR methods as opposed to the traditional ones.

Acknowledgment

This work was supported by the Spanish Ministerio de Ciencia y Tecnología under project TIC2003-00877 (including EDRF fundings).

Bibliography

- [1] E. H. Aarts and J. Korst, *Simulated Annealing and Boltzmann Machines: A Stochastic Approach to Combinatorial Optimization and Neural Computing*, John Wiley & Sons, New York, NY, USA, 1989.
- [2] P. K. Allen, A. Troccoli, B. Smith, S. Murray, I. Stamos, and M. Leordeanu, “New methods for digital modeling of historic sites,” *IEEE Computer Graphics and Applications*, vol. 23, no. 6, pp. 32–41, 2003.
- [3] K. S. Arun, T. S. Huang, and S. D. Blostein, “Least-squares fitting of two 3-D point sets,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 9, no. 5, pp. 698–700, 1987.

- [4] M. A. Audette, F. P. Ferrie, and T. M. Peters, "An algorithmic overview of surface registration techniques for medical imaging," *Medical Image Analysis*, vol. 4, no. 3, pp. 201–217, 2000.
- [5] T. Bäck, D. B. Fogel, and Z. Michalewicz, Eds., *Handbook of Evolutionary Computation*, IOP, Bristol, UK; Oxford University Press, Oxford, UK, 1997.
- [6] E. Bardinet, S. Fernández Vidal, S. Damas Arroyo, G. Malandain, and N. Pérez de la Blanca Capilla, "Structural object matching," in *Proceedings of the 2nd International Symposium on Advanced Concepts of Intelligent Vision Systems (ACIVS '00)*, vol. 2, pp. 73–77, Baden-Baden, Germany, August 2000.
- [7] D. I. Barnea and H. F. Silverman, "A class of algorithms for fast digital image registration," *IEEE Transactions on Computers*, vol. 21, no. 2, pp. 179–186, 1972.
- [8] P. J. Besl and N. D. McKay, "A method for registration of 3-D shapes," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 14, no. 2, pp. 239–256, 1992.
- [9] L. G. Brown, "A survey of image registration techniques," *ACM Computing Surveys*, vol. 24, no. 4, pp. 325–376, 1992.
- [10] P. Chalermwat, T. El-Ghazawi, and J. LeMoigne, "2-phase GA-based image registration on parallel clusters," *Future Generation Computer Systems*, vol. 17, no. 4, pp. 467–476, 2001.
- [11] C. Chen and I. Stamos, "Semi-automatic range to range registration: a feature-based method," in *Proceedings of the 5th International Conference on 3-D Digital Imaging and Modeling (3DIM '05)*, pp. 254–261, Ottawa, Ontario, Canada, June 2005.
- [12] C. K. Chow, H. T. Tsui, and T. Lee, "Surface registration using a dynamic genetic algorithm," *Pattern Recognition*, vol. 37, no. 1, pp. 105–117, 2004.
- [13] C. K. Chow, H. T. Tsui, T. Lee, and T. K. Lau, "Medical image registration and model construction using genetic algorithms," in *Proceedings of the 1st International Workshop on Medical Imaging and Augmented Reality (MIAR '01)*, pp. 174–179, Sha Tin New Town, Hong Kong, June 2001.
- [14] D. L. Collins, A. P. Zijdenbos, V. Kollokian, et al., "Design and construction of a realistic digital brain phantom," *IEEE Transactions on Medical Imaging*, vol. 17, no. 3, pp. 463–468, 1998.
- [15] O. Cordón and S. Damas, "Image registration with iterated local search," *Journal of Heuristics*, vol. 12, no. 1-2, pp. 73–94, 2006.
- [16] O. Cordón, S. Damas, and J. Santamaría, "Feature-based image registration by means of the CHC evolutionary algorithm," *Image and Vision Computing*, vol. 24, no. 5, pp. 525–533, 2006.
- [17] O. Cordón, S. Damas, and J. Santamaría, "A fast and accurate approach for 3D image registration using the scatter search evolutionary algorithm," *Pattern Recognition Letters*, vol. 27, no. 11, pp. 1191–1200, 2006.
- [18] O. Cordón, S. Damas, J. Santamaría, and R. Martí, "Scatter search for the point matching problem in 3D image registration," to appear in *INFORMS Journal on Computing*.
- [19] K. Deb and D. E. Goldberg, "An investigation of niche and species formation in genetic function optimization," in *Proceedings of the 3rd International Conference on Genetic Algorithms and Their Application (ICGA '89)*, J. D. Schaffer, Ed., pp. 42–50, Fairfax, Va, USA, June 1989.
- [20] E. De Castro and C. Morandi, "Registration of translated and rotated images using finite Fourier transforms," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 9, no. 5, pp. 700–703, 1987.
- [21] S. F. El-Hakim and H. Ziemann, "A step-by-step strategy for gross-error detection," *Photogrammetric Engineering & Remote Sensing*, vol. 50, no. 6, pp. 713–718, 1984.
- [22] L. J. Eshelman, "The CHC adaptive search algorithm: how to have safe search when engaging in nontraditional genetic recombination," in *Proceedings of the 1st Workshop on Foundations of Genetic Algorithms*, G. J. E. Rawlins, Ed., pp. 265–283, Bloomington, Ind, USA, July 1991.
- [23] O. Faugeras, *Three-Dimensional Computer Vision*, MIT Press, Cambridge, Mass, USA, 1996.
- [24] J. Feldmar and N. Ayache, "Rigid, affine and locally affine registration of free-form surfaces," *International Journal of Computer Vision*, vol. 18, no. 2, pp. 99–119, 1996.
- [25] J. M. Fitzpatrick, J. J. Grefenstette, and D. Van Gucht, "Image registration by genetic search," in *Proceedings of IEEE Southeastcon Conference*, pp. 460–464, Louisville, Ky, USA, April 1984.
- [26] W. Förstner, "The reliability of block triangulation," *Photogrammetric Engineering & Remote Sensing*, vol. 51, no. 8, pp. 1137–1149, 1985.

- [27] D. E. Goldberg, *Genetic Algorithms in Search, Optimization and Machine Learning*, Addison-Wesley, Boston, Mass, USA, 1989.
- [28] D. E. Goldberg and J. Richardson, "Genetic algorithms with sharing for multimodal function optimization," in *Proceedings of the 2nd International Conference on Genetic Algorithms on Genetic Algorithms and Their Application (ICGA '87)*, pp. 41–49, Cambridge, Mass, USA, July 1987.
- [29] A. A. Goshtasby, *2-D and 3-D Image Registration for Medical, Remote Sensing, and Industrial Applications*, John Wiley & Sons, New York, NY, USA, 2005.
- [30] R. He and P. A. Narayana, "Global optimization of mutual information: application to three-dimensional retrospective registration of magnetic resonance images," *Computerized Medical Imaging and Graphics*, vol. 26, no. 4, pp. 277–292, 2002.
- [31] M. Held, W. Weiser, and F. Wilhelmstötter, "Fully automatic elastic registration of MR images with statistical feature extraction," *Journal of WSCG*, vol. 12, no. 1, pp. 153–160, 2004.
- [32] F. Herrera, M. Lozano, and J. L. Verdegay, "Tackling real-coded genetic algorithms: operators and tools for behavioural analysis," *Artificial Intelligence Review*, vol. 12, no. 4, pp. 265–319, 1998.
- [33] J. H. Holland, *Adaptation in Natural and Artificial Systems*, MIT Press, Cambridge, Mass, USA, 1975.
- [34] B. K. P. Horn, "Closed-form solution of absolute orientation using unit quaternions," *Journal of the Optical Society of America A*, vol. 4, no. 4, pp. 629–642, 1987.
- [35] P. J. Huber, *Robust Statistics*, John Wiley & Sons, New York, NY, USA, 1981.
- [36] J. Juhl, "Part I: det interaktive on-line program oANA. Part II: det analytiske straleudjaevningsprogram sANA," Aalborg University Centre, Aalborg, Denmark 1980.
- [37] R. K.-S. Kwan, A. C. Evans, and B. Pike, "MRI simulation-based evaluation of image-processing and classification methods," *IEEE Transactions on Medical Imaging*, vol. 18, no. 11, pp. 1085–1097, 1999.
- [38] M. Laguna and R. Martí, *Scatter Search: Methodology and Implementations in C*, Kluwer Academic Publishers, Dordrecht, The Netherlands, 2003.
- [39] Y. Liu, "Improving ICP with easy implementation for free-form surface matching," *Pattern Recognition*, vol. 37, no. 2, pp. 211–226, 2004.
- [40] J. Luck, C. Little, and W. Hoff, "Registration of range data using a hybrid simulated annealing and iterative closest point algorithm," in *Proceedings of IEEE International Conference on Robotics and Automation (ICRA '00)*, vol. 4, pp. 3739–3744, San Francisco, Calif, USA, April 2000.
- [41] D. G. Luenberger, *Optimization by Vector Space Methods*, John Wiley & Sons, New York, NY, USA, 1969.
- [42] F. Maes, D. Vandermeulen, and P. Suetens, "Comparative evaluation of multiresolution optimization strategies for multimodality image registration by maximization of mutual information," *Medical Image Analysis*, vol. 3, no. 4, pp. 373–386, 1999.
- [43] S. W. Mahfoud, "Niching methods for genetic algorithms," Tech. Rep. IlliGAL-95001, Genetic Algorithms Laboratory, University of Illinois at Urbana-Champaign, Urbana, Ill, USA, 1995.
- [44] V. R. Mandava, J. M. Fitzpatrick, and D. R. Pickens, "Adaptive search space scaling in digital image registration," *IEEE Transactions on Medical Imaging*, vol. 8, no. 3, pp. 251–262, 1989.
- [45] G. K. Matsopoulos, N. A. Mouravliansky, K. K. Delibasis, and K. S. Nikita, "Automatic retinal image registration scheme using global optimization techniques," *IEEE Transactions on Information Technology in Biomedicine*, vol. 3, no. 1, pp. 47–60, 1999.
- [46] O. Monga, S. Benayoun, and O. D. Faugeras, "From partial derivatives of 3-D density images to ridge lines," in *Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR '92)*, pp. 354–359, Champaign, Ill, USA, June 1992.
- [47] D. M. Muratore, J. H. Russ, B. M. Dawant, and R. L. Galloway Jr., "Three-dimensional image registration of phantom vertebrae for image-guided surgery: a preliminary study," *Computer Aided Surgery*, vol. 7, no. 6, pp. 342–352, 2002.
- [48] P. Rogelj, S. Kovacic, and J. C. Gee, "Validation of a non-rigid registration algorithm for multimodal data," in *Medical Imaging*, M. Sonka and J. M. Fitzpatrick, Eds., vol. 4684 of *Proceedings of SPIE*, pp. 299–307, San Diego, Calif, USA, February 2002.
- [49] J.-M. Rouet, J.-J. Jacq, and C. Roux, "Genetic algorithms for a robust 3-D MR-CT registration," *IEEE Transactions on Information Technology in Biomedicine*, vol. 4, no. 2, pp. 126–136, 2000.

- [50] K. Shoemake, "Animating rotation with quaternion curves," in *Proceedings of the 12th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '85)*, pp. 245–254, San Francisco, Calif, USA, July 1985.
- [51] K. Simunic and S. Loncaric, "A genetic search-based partial image matching," in *Proceedings of the 2nd IEEE International Conference on Intelligent Processing Systems (ICIPS '98)*, pp. 119–122, Gold Coast, Australia, August 1998.
- [52] M. Svedlow, C. D. McGilllem, and P. E. Anuta, "Experimental examination of similarity measures and preprocessing methods used for image registration," in *Proceedings of Symposium on Machine Processing of Remotely Sensed Data*, pp. 4a–9–17, West Lafayette, Ind, USA, June–July 1976.
- [53] P. W. M. Tsang, "A genetic algorithm for aligning object shapes," *Image and Vision Computing*, vol. 15, no. 11, pp. 819–831, 1997.
- [54] P. Viola and W. M. Wells III, "Alignment by maximization of mutual information," *International Journal of Computer Vision*, vol. 24, no. 2, pp. 137–154, 1997.
- [55] M. P. Wachowiak, R. Smolíková, Y. Zheng, J. M. Zurada, and A. S. Elmaghhraby, "An approach to multimodal biomedical image registration utilizing particle swarm optimization," *IEEE Transactions on Evolutionary Computation*, vol. 8, no. 3, pp. 289–301, 2004.
- [56] S. M. Yamany, M. N. Ahmed, and A. A. Farag, "A new genetic-based technique for matching 3-D curves and surfaces," *Pattern Recognition*, vol. 32, no. 10, pp. 1817–1820, 1999.
- [57] S. Y. Yuen, C. K. Fong, and H. S. Lam, "Guaranteeing the probability of success using repeated runs of genetic algorithm," *Image and Vision Computing*, vol. 19, no. 8, pp. 551–560, 2001.
- [58] Z. Zhang, "Iterative point matching for registration of free-form curves and surfaces," *International Journal of Computer Vision*, vol. 13, no. 2, pp. 119–152, 1994.
- [59] B. Zitová and J. Flusser, "Image registration methods: a survey," *Image and Vision Computing*, vol. 21, no. 11, pp. 977–1000, 2003.

Oscar Cordón: European Centre for Soft Computing, 33600 Mieres, Asturias, Spain; Department of Computer Science and Artificial Intelligence, University of Granada, 18071 Granada, Spain

Email: ocordon@decsai.ugr.es

Sergio Damas: European Centre for Soft Computing, 33600 Mieres, Asturias, Spain; Software Engineering Department, Computer Engineering School, University of Granada, 18071 Granada, Spain

Email: sdamas@ugr.es

José Santamaría: Department of Software Engineering, University of Cádiz, 11002 Cádiz, Spain

Email: jose.santamarialopez@uca.es

13

Image segmentation hybridizing variable neighborhood search and memetic algorithms

Abraham Duarte, Ángel Sánchez,
Felipe Fernández, and
Antonio S. Montemayor

13.1. Introduction

Image segmentation consists of subdividing an image into its constituent regions or objects [10]. The level of subdivision depends on the specific problem being solved. The segmentation result is the labeling of the pixels in the image with a small number of labels. This partition is accomplished in such a way that the pixels belonging to homogeneous regions regarding to one or more features (i.e., brightness, texture, or color) share the same label, and regions of pixels with significantly different features have different labels.

According to Ho and Lee [11], four objectives must be considered for developing an efficient generalized segmentation algorithm: continuous closed contours, nonoversegmentation, independence of threshold setting, and short computation time. Many segmentation approaches have been proposed in the literature [10, 21, 25]. Roughly speaking, they can be classified as edge-based, threshold-based, and region-based methods. In this chapter, a method is presented which can be considered as region-based and it pursues a high-level extraction of the image structures. As a result, the method produces a k -region partition of the scene. We take into account this approach by representing an oversegmented version of an original image as an undirected weighted graph. In this graph, nodes are the image regions and the edges together with their associated weights are defined using local information. A high-quality k -partition that uses a variant of min-cut value (normalized cut [24]) for the image graph is computed. The application of a low-level [26] hybridization between variable neighborhood search (VNS) [15, 18] and a memetic algorithm [19] to efficiently solve the image segmentation problem is the core of the proposed method.

Many optimization problems are too difficult to be solved exactly in a reasonable amount of time. Due to the complexity of these problems, efficient approximate solutions may be preferable in practical applications. Heuristic algorithms are proposed in this direction. Examples of heuristics are many local search procedures that are problem specific and do not guarantee the optimality.

Metaheuristics are high-level general strategies for designing heuristics procedures [9, 16, 29]. The relevance of metaheuristics is reflected in their application for acceptably solving many different real-world complex problems, mainly combinatorial. Since the initial proposal of Glover about Tabu search in 1986, many metaheuristics have emerged to design good general heuristics methods for solving different domain application problems. Genetic programming [17], GRASP [8], simulated annealing [1], or ant colony optimization [6] are other well-known examples of metaheuristics. Their relevance is reflected in the publication of many books and papers on this research area [9, 29].

The application of evolutionary techniques to image processing and computer vision problems has increased mainly due to the robustness of the approach [21]. Many image analysis tasks like image enhancement, feature selection, and image segmentation have been effectively solved using genetic programming [22]. Among these tasks, segmentation is one of the most difficult ones. Usually, the standard linear segmentation methods are insufficient for a reliable object classification.

The usage of some nonlinear approaches like neural networks or mathematical morphology methods has provided better results [25]. However, the inherent complexity of many scenes (i.e., images with noncontrolled illumination conditions or textured images) makes it very difficult to achieve an optimal pixel classification into regions, due to the combinatorial nature of the task.

Evolutionary image segmentation [11, 22, 31] has reported a good performance in relation to more classical segmentation methods. Our approach of modeling and solving image segmentation as a graph partitioning problem is related to Shi and Malik's work [24]. They use a computational technique based on a generalized eigenvalue problem for computing the segmented regions.

This chapter introduces a hybrid evolutionary algorithm as a graph-based efficient segmentation technique to improve quality results. This new algorithm is based on a low-level hybridization between a variable neighborhood search (VNS) and a memetic algorithm (MA). In Section 13.2, several approaches, related to our work, are revised. Section 13.3 resumes a brief introduction to memetic algorithms. The optimization strategy is introduced in Section 13.4. VNS metaheuristic is presented in Section 13.5. Section 13.6 describes our evolutionary algorithm proposal. Graph construction details and method overview are described in Section 13.7. In Section 13.8, some experimental results are presented. Finally, we conclude this chapter in Section 13.9.

13.2. Related work

This section revises two segmentation approaches related to our work: metaheuristics-based and graph-cut-based. The first approach consists of considering the segmentation task as an optimization problem in which an objective function is improved by means of a metaheuristic procedure. The second approach consists of transforming the image into a graph, where some graph cut techniques are applied.

13.2.1. Metaheuristics review in image segmentation framework

Metaheuristics are general procedures successfully applied to a large diversity of very hard combinatorial optimization problems. Surprisingly, compared to the amount of research undertaken on these optimization problems, relatively little work has been devoted to the application of metaheuristics to computer vision and image processing, despite the potential advantages of robustness, quality, and efficiency [21].

Many image analysis tasks like image enhancement, feature selection, and image segmentation may be effectively solved using metaheuristics [22]. Among these tasks, segmentation is, in general, one of the most difficult tasks. Usually the standard linear segmentation methods are insufficient for a reliable object classification.

The inherent complexity of many scenes (i.e., images with noncontrolled illumination conditions or textured images) makes very difficult to achieve an optimal pixel classification into regions, due to the combinatorial nature of the task.

Metaheuristics-based segmentation has been focused on the use of evolutionary algorithms [11, 22, 31] that have reported a good performance in relation to more classical segmentation methods. A reduced number of papers using other metaheuristics for image segmentation have been reported. In general, unsupervised image segmentation is modeled as a clustering problem, which has been tackled using fuzzy algorithms [2, 13] and ant colony optimization (ACO) metaheuristic [20].

Our approach for modeling and solving image segmentation as a graph partitioning problem is related to Shi and Malik's proposal [24]. These authors use a computational technique based on a generalized eigenvalue problem for computing the segmentation resulting regions.

13.2.2. Image segmentation via graph cuts

The first work in image graph-based segmentation method used fixed neighborhood structures and local measures in computing segmentation [32]. In [4] a method is presented based on the computing of the minimum spanning tree of the image graph, and it has also been successfully used in clustering applications. In general, a high-quality segmentation is obtained for simple images (i.e., synthetic) but, for complex images, the results are not acceptable. In [27] an edge weight normalization stage is proposed, which is not suitable to provide a reasonable adaptive segmentation results.

Recent literature has witnessed two popular graph cut segmentation methods: the minimum cut (and their variants) using graph cuts analysis [24, 28, 30] and energy minimization, using the max flow algorithm [14, 23]. More recently, a third major approach has been proposed based on a generalization of Swendsen-Wang method [3]. In this chapter, we focus on a variant of min-cut (normalized cut [24]) approach because this formulation has achieved successful results in image segmentation frameworks [24].

The min-cut optimization problem, defined for a weighted undirected graph $S = \{V, E, W\}$, consists of finding a bipartition G of the set of vertices or nodes of the graph: $V = (C, C')$, where $V = C \cup C'$, such that the sum of the weights of edges with endpoints in different subset is minimized. Every bipartition of the set of vertices V into C and C' is usually called a cut or cutset and the sum of the weights of the edges, with a vertex in C and the other vertex in C' , is called cut weight or similarity (s) between C and C' . For the considered min-cut optimization problem, the cut weight or similarity between C and C' , given by

$$s(C, C') = \sum_{v \in C, u \in C'} w_{vu}, \quad (13.1)$$

is minimized, where w_{vu} is the edge weight between nodes $v, u \in V$. In [12] it is demonstrated that the decision version (reformulation of the problem with binary variables) of max-cut (dual version of min-cut problem) is NP-complete. This way, we need to use approximate algorithms for finding a high-quality solution in a reasonable time. The min-cut approach has been used by Wu and Leahy [30] as a clustering method and applied to image segmentation. These authors search a partition of the image graph into k subgraphs such that the similarity (min-cut) among subgraphs is minimized. They pointed out that although for some images the segmentation result is acceptable, in general, this method produces an over-segmentation because small regions are favored. To avoid this fact other functions that try to minimize the effect of this problem are proposed [4]. The function that must be optimized (minimized) called min-max cut is

$$\text{cut}(G) = \frac{\sum_{v \in C, u \in C'} w_{vu}}{\sum_{v \in C, u \in C} w_{vu}} + \frac{\sum_{v \in C', u \in C} w_{vu}}{\sum_{v \in C', u \in C'} w_{vu}} = \frac{s(C, C')}{s(C, C)} + \frac{s(C', C')}{s(C', C')}, \quad (13.2)$$

where the numerators of this expression are the same $s(C, C')$ and the denominators are the sum of the edge weights belonging to C or C' , respectively. It is important to remark that in an image segmentation framework, it is necessary to minimize the similarity between C and C' (numerators) and maximize the similarity inside C , and inside C' (denominators). In this case, the sum of edges between C and C' is minimized, and simultaneously the sums of weights inside of each subset are maximized. In [24] an alternative cut value called normalized cut is proposed which, in general, gives better results in practical image segmentation problems:

$$\text{Ncut}(G) = \frac{\sum_{v \in C, u \in C'} w_{vu}}{\sum_{v \in C, u \in V} w_{vu}} + \frac{\sum_{v \in C', u \in C} w_{vu}}{\sum_{v \in C', u \in V} w_{vu}} = \frac{s(C, C')}{s(C, V)} + \frac{s(C', C')}{s(C', V)}. \quad (13.3)$$

13.3. Memetic algorithms

Unlike traditional genetic algorithms (GA), memetic algorithms (MA) [19] are intrinsically concerned with exploiting available knowledge about the problem under study. This approach is not an optimal mechanism but, in general, yields to a

solution improvement. Optimization is accomplished in MA framework by incorporating problem dependent heuristics: approximation algorithms, local search techniques, specialized recombination operators, and so forth. Moreover, MAs can be additionally improved by means of a low-level or high-level hybridization [26] with other metaheuristics.

MAs are a search strategy in which a population of optimizing individuals (called optimizing agents in MA context) cooperate and compete in order to get high-quality solutions. Cooperative-competitive strategy of optimizing agents get a synergy among the different search approach incorporated. The most distinctive characteristic of MAs is the inclusion of problem knowledge which is supported by no-free-lunch theorem which establishes that the quality and robustness of one algorithm is in accordance with the amount of information that they incorporate in its own design.

In order to design an MA, the main template is inherited from GAs and only the problem specific details have to be rewritten, such as the definition of problem-dependent recombination and mutation operators, the population initialization function and the local search. Notice that any kind of the problem knowledge can be incorporated creating new memetic operators. In summary, the following steps are thus necessary for designing an MA.

- (1) Find a suitable representation for the solutions (individuals) and an evaluation function for calculating the fitness of a given solution based on the referred representation.
- (2) Find a local search strategy.
- (3) Find a suitable initialization method for the initial population.
- (4) Define a problem-dependent mutation and recombination operators.

13.4. Optimization strategy

There are some metaheuristics which are mainly concerned with diversification purposes. For instance, evolutionary algorithms (EA) belong to this kind of procedures. On the other hand, another metaheuristic such as variable neighborhood search (VNS) and their variants [15, 18] focuses almost entirely on the search intensification. Regarding this fact, EA and VNS can be considered as complementary algorithms. In addition, the hybridization of these techniques can yield to very effective and robust methods. This chapter proposes a new evolutionary algorithm based on a low-level hybridization [26] between a specific memetic algorithm and variable neighborhood search procedure. The developed algorithm is a good tradeoff between intensification and diversification strategies.

The intensification phase is mainly carried out by the VNS procedure. This metaheuristic intensively looks for quality solutions in a predefined set of neighborhood structures. If the search procedure is got stuck, VNS changes the neighborhood structure in order to get away from local optimum. Notice that although the main task of VNS is the search intensification, this metaheuristic also diversifies the search procedure by means of neighborhood changing. On the other hand,

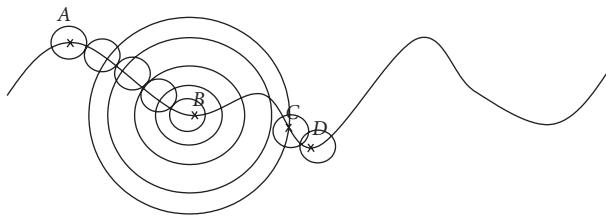


FIGURE 13.1. VNS search process.

EA objective is mainly related with the diversification stage. This task is accomplished with traditional operators (selection, mutation, and crossover) enriched with some knowledge about the problem. Notice that although the main task of MA is the search diversification, this metaheuristic also intensifies the search procedure by means of population evolution and the inclusion of problem-dependent operators. In the following two sections, we, respectively, describe the VNS and the hybrid MA implementations, developed for the segmentation problem.

13.5. Variable neighborhood search

This section resumes the main features of variable neighborhood search (VNS) metaheuristic. This metaheuristic, which was originally proposed by Hansen and Mladenović [15, 18], is based on the exploration of a dynamic neighborhood model. Each step has three major phases: neighbor generation, local search, and jump.

Unlike other metaheuristics based on local search methods, VNS allows changes of the neighborhood structure during the search. The basic idea of VNS is to change the neighborhood structure when the local search is trapped on a local optimum. In Figure 13.1 a landscape example of a given objective function is presented. The search process starts from A and, after applying successively the search procedure, it was trapped in the local optimum B . The metaheuristic VNS by means of a systematic change of the neighborhood structure (augmenting the considered neighborhood) could escape from B , following with the search procedure and reaching C . Then, the neighborhood structure returns to its original size and VNS goes on with the search, reaching the following local optimum D . This process is maintained until a determined finish condition is met.

Let N_k , $k = 1, \dots, k_{\max}$, be a set of predefined neighborhood structures and let $N_k(x)$ be the set of solutions in the k -order neighborhood of a solution x . In the first phase, a neighbor $x' \in N_k(x)$ of the current solution is applied. Next, a solution x'' is obtained by applying local search to x' . Finally, the current solution jumps from x to x'' if x'' is a better solution than x . Otherwise, the neighborhood order k is increased by one and the above steps are repeated until some stopping condition is met. The pseudocode of a typical VNS procedure is illustrated in Algorithm 13.1.

```

Procedure VNS (x)
  var
    x: Initial solution
    x', x'': Intermediate solutions
    k: Neighbourhood order
  begin
    /*First Neighbourhood Structure*/
    k = 1;
    while k < kmax do
      /*Select an random solution in k-
      neighbourhood structure*/
      x' = Random (x, Nk(x))
      /* Use the local search procedure shown
      in Figure 7.3*/
      x'' = LocalSearch (x');
      /* Replace the actual solution by the new
      one when an improvement is obtained */
      if w(x'') > w(x) then
        x = x'';
        k = 1;
      else
        k = k + 1;
      end if
    end while
  end
end VNS

```

ALGORITHM 13.1. VNS high-level pseudocode.

In the segmentation framework described in this chapter, the k -order neighborhood is defined by all solutions that can be derived from the current one by selecting k vertices from one subset of the vertex bipartition and transferring them to the other subset.

The local search phase is based on the following neighborhood structure. Let (C_a, C'_a) be the current cutset solution. For each vertex $v \in V$ we associate a new neighbor cutset (C_b, C'_b) :

$$(C_b, C'_b) = N_1(C_a, C'_a) = \begin{cases} C_b = C_a - \{v\}; C'_b = C'_a + \{v\} & \text{if } v \in C_a, \\ C_b = C_a + \{v\}; C'_b = C'_a - \{v\} & \text{if } v \in C'_a. \end{cases} \quad (13.4)$$

We define for each node $v \in V$ the functions C_to_C' and C'_to_C as

$$\begin{aligned} C_to_C'(v) &= \frac{s(C, C') + \sigma'(v) - \sigma(v)}{s(C, V) + \sigma(v)} + \frac{s(C, C') + \sigma'(v) - \sigma(v)}{s(C', V) - \sigma'(v)}, \\ C'_to_C(v) &= \frac{s(C, C') - \sigma'(v) + \sigma(v)}{s(C, V) - \sigma'(v)} + \frac{s(C, C') - \sigma'(v) + \sigma(v)}{s(C', V) + \sigma(v)}, \end{aligned} \quad (13.5)$$

```

Procedure Local_Search (g)
  var
    g = (C, C'): Cutset structure
  begin
    for v = 1 to Nodes_in_considered_graph do
      if v ∈ C and C_to_C'(v) > 0 then
        /* v : C → C' */
        C = C \ {v};
        C' = C' ∪ {v};
      end if
      if v ∈ C' and C'_to_C(v) > 0 then
        /* v : C' → C */
        C' = C' \ {v};
        C = C ∪ {v};
      end if
    end for
  end
end Local_Search

```

ALGORITHM 13.2. Local search high-level pseudocode.

where $\sigma(v)$ and $\sigma'(v)$ are

$$\sigma(v) = \sum_{u \in C} w_{vu}, \quad \sigma'(v) = \sum_{u \in C'} w_{vu}. \quad (13.6)$$

These two functions (C_to_C' and C'_to_C) are characterized by the change in the objective function value (fitness) associated with moving vertex v from one subset of the cut to the other. These two functions are highly related with $Ncut$ function, defined in Section 13.2.

In order to improve the objective function value, a vertex makes a movement in the two following situations:

$$\begin{aligned} \text{if } v \in C \wedge C_to_C'(v) > 0, & \quad \text{then } C \rightarrow C', \\ \text{if } v \in C' \wedge C'_to_C(v) > 0, & \quad \text{then } C' \rightarrow C, \end{aligned} \quad (13.7)$$

where $C \rightarrow C'$ (equivalently $C' \rightarrow C$) represents the movement of vertex v from subset C to C' ($C \cup C' = V$).

All possible moves are examined. The current solution is replaced by its best improving neighbor solution. The search stops after all possible moves have been evaluated and no improving neighbor is found. The used local search strategy is summarized by the pseudocode of Algorithm 13.2.

This local search procedure tests all possible movements for each node between C and C' and vice versa. Therefore, the current solution is replaced by the best solution found in the neighborhood structure defined above. The procedure ends when no possible neighbor movement improves the current solution.

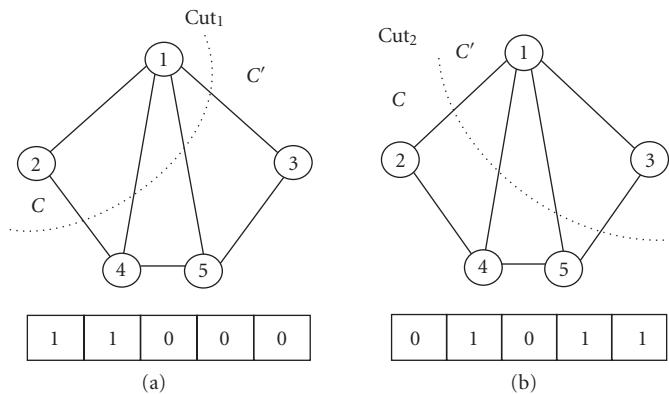


FIGURE 13.2. Cutsets representation.

13.6. Hybrid metaheuristic

This section describes a new evolutionary low-level hybridization for the image segmentation problem. In order to use a memetic algorithm for solving this problem, we need to code each feasible solution. Let $V = \{1, \dots, n\}$ be the nodes set of a given graph. The possible cuts on this graph can be coded by a Boolean n -vector $I = (i_1, \dots, i_n)$ such that the value of each component $i_u \in \{0, 1\}$ with $1 \leq u \leq n$ is given by the characteristic function

$$i_u = \begin{cases} 1 & \text{if } u \in C, \\ 0 & \text{if } u \in C'. \end{cases} \quad (13.8)$$

Figures 13.2(a) and 13.2(b) show two examples of cuts and their respective encodings.

For selecting high-quality individuals, we used a fitness-proportionate selection [16, 17], which favors individuals with high-fitness value without suppressing the chance of selection of individuals with low fitness, thus avoiding premature convergence of the population.

The proposed algorithm starts with a random initial population of individuals which represent cuts, generated by *InitialPopulation* procedure. Then, these cuts are improved (with probability P_i) by means of a local search procedure described in Algorithm 13.2.

A subset of individuals are selected using a fitness-proportionate selection. Some selected individuals are crossed over, with a probability P_r . In the proposed implementation, we have not used standard crossover (breaking individuals (parents) in several parts and generating the offspring merging parts of different parents) because this method can destroy the high-quality structures obtained by means of evolution. We have considered a fixed crossover method [5, 19], which takes into account the structural information of each individual and provides, in general, better offspring [19]. Specifically, fixed crossover is implemented as

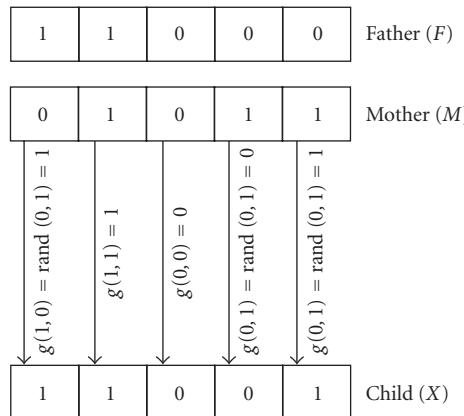


FIGURE 13.3. Fixed crossover procedure.

follows: given a father $F = \{f_1, \dots, f_n\}$ and a mother $M = \{m_1, \dots, m_n\}$, their corresponding child $X = \{x_1, \dots, x_n\}$ is determined by the considered fixed crossover function $g : \{1, 0\} \times \{1, 0\} \rightarrow \{1, 0\}$. With this crossover function, each bit x_u of new offspring is given by the following random Boolean function:

$$x_u = g(f_u, m_u) = \begin{cases} 0 & \text{if } (f_u = 0) \wedge (m_u = 0), \\ 1 & \text{if } (f_u = 1) \wedge (m_u = 1), \\ \text{rand}(0, 1) & \text{if } f_u \neq m_u, \end{cases} \quad (13.9)$$

where $\text{rand}(0, 1)$ is a random Boolean value ($\{0, 1\}$). In this way, if both parents are in the same subset, the offspring node lies in this subset. Otherwise, the node is randomly assigned to one of the subsets. Graphically, the crossover strategy is presented in Figure 13.3.

To end up the evolution cycle, new individuals are subject to mutation (a random change of a node from C to C' or vice versa) with probability $P_m = 1/|V|$. By this way, the allele mutation probability (P_m) is problem independent.

Algorithm 13.3 shows the high-level pseudocode of the corresponding hybrid evolutionary algorithm.

13.7. Method overview

Pixel-based and region-based evolutionary segmentation approaches share the same basic structure. The only difference between the two methods is that region-based methods produce an oversegmented image, as a preprocessing stage, for example, applying standard watershed segmentation to the initial brightness image. So, pixel-based methods use as input image the original image and region-based methods use as input image the oversegmented image.

In region-based approach a method constructs the corresponding weighted graph for the input image (the previously oversegmented image). This graph is

```

Procedure Hybrid_Evolutionary_Algorithm (g)
  var
    g = (C, C'): individual cutset structure
    pop: population of cutsets
    MaxGen: Number of Generations
    PopSize: Number of individuals
    pc, pm: Cross. and mut. probabilities
    pi: Improvement probability
    i: Generation Counter
    j: individual Counter
  begin
    /*Generate random cuts individuals*/
    pop = Initial_Population (g);
    /*Optimize initial population*/
    pop = Apply (Local_Search (), pi); /*Figure 7.3*/
    Best_Solution = Evaluate_Population (pop); /*Ncut*/
    for i = 1 to MaxGen do
      j = 0;
      while (j < PopulationSize) do
        /*Criteria: Random Wheel*/
        father = Selection (pop); /*Ncut*/
        r = rand01 (); /*Random function*/
        if (r < pc) then
          /*Criteria: Random Wheel*/
          mother = Selection (pop); /*Ncut*/
          child = FixCross (father, mother);
          Apply (VNS (child), pi); /*Figure 7.2*/
          pop = InsertInPopulation (pop, child);
          j = j + 1;
        end if
      end while
      pop = Apply (Mutation (pop), pm);
      Best_Solution = Evaluate_Population (pop); /*Ncut*/
    end for
    return Best_Solution
  end Hybrid_Evolutionary_Algorithm ()

```

ALGORITHM 13.3. Hybrid algorithm high-level pseudocode.

defined by representing each resulting region by a unique node and defining the edges and corresponding edge weights as a measure of spatial location, grey level average difference, and cardinality between the corresponding regions.

The oversegmented image may be modeled by means of the region adjacency graph (RAG), a usual data structure for representing region neighborhood relations in a segmented image [25]. In this graph, adjacent regions are merged in order to reduce the number of regions until a semantically meaningful segmentation is obtained.

Our approach shares this perspective and provides as segmentation result an adaptable tree-based image bipartition where the first levels of decomposition correspond to major areas or objects in the segmented image. With this aim, we propose a new data structure, called modified region adjacency graph [7] (MRAG)

that takes advantages of both RAG and pixel-based representations in which non-adjacent pixels can be joined. The MRAG structure is an undirected weighted graph $G = \{V, E, W\}$, where the set of nodes (V) represents the set of centers of gravity of each oversegmented region, and the set of weighted edges can correspond to nonadjacent regions.

The edge weights $w_{ij} \in W$ are computed by the following function:

$$w_{ij} = \begin{cases} e^{-C_{ij}(I_i - I_j)^2/\sigma_I^2} \cdot e^{-C_{ij}(x_i - x_j)^2/\sigma_x^2}, & |x_i - x_j| < r_x, \\ 0, & \text{elsewhere,} \end{cases} \quad (13.10)$$

where r_x is an experimental threshold value, I_i is the grey level mean intensity of region i , and x_i is its center of gravity. The values of σ_I , σ_x , and r_x are adjusted experimentally and, in general, they depend on the image features. Nonsignificant weighted edges, according to defined similarity criteria, are removed from the image graph. Finally, C_{ij} takes into account the cardinality of the regions i and j . This value is given by

$$C_{ij} = \frac{\|E_i\| \|E_j\|}{\|E_i\| + \|E_j\|}, \quad (13.11)$$

where $\|E_i\|$, $\|E_j\|$ are, respectively, the number of pixels in regions i and j .

The final stage of the process consists of iteratively applying the considered algorithm in a hierarchical fashion to the corresponding subgraph, resulting from the previous graph bipartition, until a termination condition is met. This condition is a tradeoff between a required segmentation precision and efficiency. This stage itself constitutes an effective region merging method for oversegmented images.

13.8. Experimental results

This section describes the experimental results obtained using the proposed hybrid metaheuristic. Experiments were performed in an Intel Pentium 4 processor at 1.7 GHz, with 256 MB of RAM. All algorithms were coded in C++, without optimization, and by the same programmer in order to have more comparable results.

The main parameters and corresponding used values of the designed hybrid evolutionary metaheuristic are as follows.

- (i) Memetic algorithm
 - (a) Initial random population of 50 individuals, called *PopSize*.
 - (b) *PopSize* is initially improved by means of the aforementioned search strategy (in Algorithm 13.2) with a probability $P_i = 0.25$.
 - (c) The probability of crossover P_c is 0.6 and it is performed by *fixed-crossover* method.
 - (d) The maximum number of generations *MaxGen* is 50.
 - (e) After the crossover, the new individuals are also improved by the described VNS strategy with a probability $P_i = 0.25$.



FIGURE 13.4. (a) Original peppers image (256×256) and (b) its corresponding oversegmented watershed transformation.

- (f) In each generation, a mutation process is applied with a probability $P_m = 1/|V|$.
- (g) The procedure ends when no individual improves its fitness or it is reached the *MaxGen* value.
- (ii) Variable neighborhood search
 - (a) Each child obtained after a fixed-crossover application is the *initial solution* for VNS procedure.
 - (b) The maximum neighborhood order k_{\max} is set to 1% of the number of nodes in the graph.

Figure 13.4(a) shows the original peppers image while Figure 13.4(b) its corresponding watershed transformation. As explained in the previous section, the method needs the image graph representation in order to apply the bipartition scheme. This graph is constructed via a standard watershed transformed image. Figure 13.5 shows the segmentation sequence for the peppers image. Note that the left image of every row carries more semantical information and is bipartitioned in a hierarchical fashion.

Figures 13.6 and 13.7 show the first iteration of the bipartition process for their respective original images. Note that in the case of the elephants the sky area is very homogeneous and the watershed transformation returns large regions compared to the others images. This fact is not very useful since it tends to contaminate weighting values in the formula.

Last example of a first stage bipartition is shown in Figure 13.8. The original image of this figure has been corrupted with salt and pepper noise in those very homogeneous areas where the watershed transformation gives very large fragments compared to the ones of the important object.

13.9. Conclusions

This chapter introduces a hybrid evolutionary algorithm as a graph-based efficient segmentation technique to improve quality results. This new algorithm is based on a low-level hybridization between a variable neighborhood search (VNS) and



FIGURE 13.5. Application of the proposed method to the peppers image shown in Figure 13.4. The oversegmented image is bipartitioned in a hierarchical fashion. From top to bottom rows: First, second and third cut images. Left image in each row is hierarchically subdivided.

a memetic algorithm (MA). On the one hand, we have used VNS as an additional intensification procedure to improve the corresponding optimization process. On the other hand, an MA is mainly used to diversify the corresponding search process. Notice that this MA includes several problem-dependent data and methods.

For the image segmentation problem, a hierarchical region-based segmentation approach has been considered. An important advantage of this graph design is that some partially occluded objects, resulting in more than one nonadjacent region in the image could be wrong merged. The image segmentation problem is now equivalent to minimize the normalize cut value in the corresponding graph.

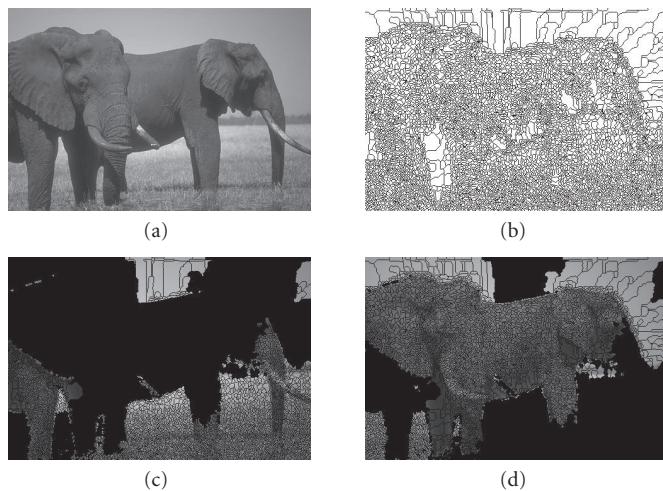


FIGURE 13.6. (a) Original image (480×320). (b) Watershed image. (c) and (d) First bipartition results.

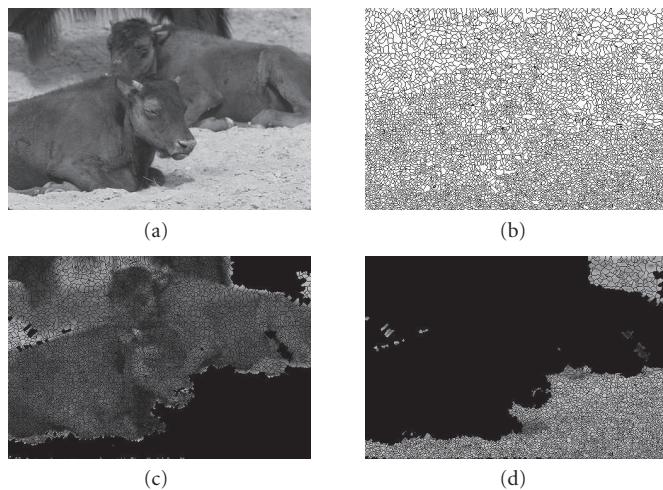


FIGURE 13.7. (a) Original image (480×320). (b) Watershed image. (c) and (d) First bipartition results.

For a region-based approach the input image is firstly oversegmented using a standard watershed algorithm. Next, the associated MRAG structure has been built to model the segmentation problem as a weighted graph.

The region representation allows processing of larger spatial resolution images than the pixel-based approach or any other typical graph-based segmentation method. Also, the hybrid algorithm provides an effective region merging method for oversegmentation problems, achieving high-quality segmentation in an efficient way. An important advantage of the approach is that MRAG structure does

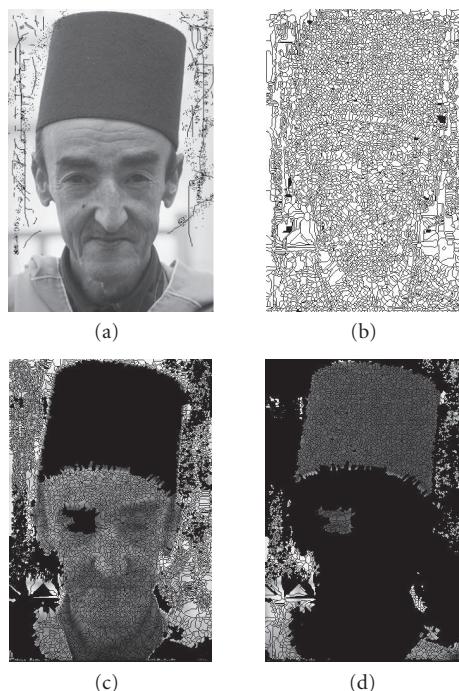


FIGURE 13.8. (a) Original image (320×480) corrupted with noise artifacts in the homogeneous areas. (b) Watershed image. (c) and (d) First bipartition results.

not need to be updated when merging regions. Moreover, the resulting hierarchical top-down segmentation degree is adaptable to the complexity of the considered image and to the application requirements. The experimental results also show that this algorithm has a robust behavior and gives high-quality solutions, independently of the graph characteristics.

Bibliography

- [1] E. H. L. Aarts and J. Korst, *Simulated Annealing and Boltzmann Machines: A Stochastic Approach to Combinatorial Optimization and Neural Computing*, John Wiley & Sons, New York, NY, USA, 1989.
- [2] L. Ballerini, L. Bocchi, and C. B. Johansson, “Image segmentation by a genetic fuzzy c-means algorithm using color and spatial information,” in *Proceedings of Applications of Evolutionary Computing*, vol. 3005 of *Lecture Notes in Computer Science*, pp. 260–269, Springer, Coimbra, Portugal, April 2004.
- [3] A. Barbu and S.-C. Zhu, “Graph partitioning by Swendsen-Wang cuts,” in *Ninth IEEE International Conference on Computer Vision (ICCV’03)*, vol. 1, pp. 320–328, 2003.
- [4] C. H. Q. Ding, X. He, H. Zha, M. Gu, and H. D. Simon, “A min-max cut algorithm for graph partitioning and data clustering,” in *Proceedings of IEEE International Conference on Data Mining (ICDM ’01)*, pp. 107–114, San Jose, Calif, USA, November–December 2001.

- [5] O. Dolezal, T. Hofmeister, and H. Lefmann, “A comparison of approximation algorithms for the MAXCUT-problem,” Reihe CI 57/99, SFB 531, Universität Dortmund, Dortmund, Germany, 1999.
- [6] M. Dorigo, V. Maniezzo, and A. Colorni, “Ant system: optimization by a colony of cooperating agents,” *IEEE Transactions on Systems, Man, and Cybernetics, Part B*, vol. 26, no. 1, pp. 29–41, 1996.
- [7] A. Duarte, Á. Sánchez, F. Fernández, and A. Sanz, “Region merging for severe oversegmented images using a hierarchical social metaheuristic,” in *Proceedings of Applications of Evolutionary Computing*, vol. 3449 of *Lecture Notes in Computer Science*, pp. 345–355, Springer, Lausanne, Switzerland, March–April 2005.
- [8] T. A. Feo and M. G. C. Resende, “Greedy adaptive search procedures,” *Journal of Global Optimization*, vol. 6, no. 2, pp. 109–133, 1995.
- [9] F. Glover and G. Kochenberger, Eds., *Handbook of Metaheuristics*, Kluwer Academic Publishers, Norwell, Mass, USA, 2003.
- [10] R. C. Gonzalez and R. E. Woods, *Digital Image Processing*, Prentice-Hall, Upper Saddle River, NJ, USA, 2nd edition, 2002.
- [11] S.-Y. Ho and K.-Z. Lee, “Design and analysis of an efficient evolutionary image segmentation algorithm,” *The Journal of VLSI Signal Processing*, vol. 35, no. 1, pp. 29–42, 2003.
- [12] R. M. Karp, “Reducibility among combinatorial problems,” in *Complexity of Computer Computations*, R. Miller and J. W. Thatcher, Eds., pp. 85–103, Plenum Press, New York, NY, USA, 1972.
- [13] S.-M. Kim and W. Kim, “An algorithm for segmenting gaseous objects on images,” in *Proceedings of Applications of Evolutionary Computing*, vol. 3005 of *Lecture Notes in Computer Science*, pp. 322–328, Springer, Coimbra, Portugal, April 2004.
- [14] V. Kolmogorov and R. Zabih, “What energy functions can be minimized via graph cuts?” in *Proceedings of the 7th European Conference on Computer Vision (ECCV '02)*, vol. 3, pp. 65–81, Copenhagen, Denmark, May 2002.
- [15] P. Hansen and N. Mladenović, “Developments of variable neighborhood search,” in *Essays and Surveys in Metaheuristics*, C. C. Ribeiro and P. Hansen, Eds., pp. 415–439, Kluwer Academic Publishers, Norwell, Mass, USA, 2001.
- [16] Z. Michalewicz and D. B. Fogel, *How to Solve It: Modern Heuristics*, Springer, Berlin, Germany, 2nd edition, 2000.
- [17] Z. Michalewicz, *Genetic Algorithms + Data Structures = Evolution Programs*, Springer, New York, NY, USA, 3rd edition, 1996.
- [18] N. Mladenović and P. Hansen, “Variable neighborhood search,” *Computers and Operations Research*, vol. 24, no. 11, pp. 1097–1100, 1997.
- [19] P. Moscato and C. Cotta, “A gentle introduction to memetic algorithms,” in *Handbook of Metaheuristics*, F. Glover and G. Kochenberger, Eds., pp. 105–144, Kluwer Academic Publishers, Boston, Mass, USA, 2003.
- [20] S. Ouadfel and M. Batouche, “Unsupervised image segmentation using a colony of cooperating ants,” in *Proceedings of the 2nd International Workshop on Biologically Motivated Computer Vision (BMCV '02)*, vol. 2525 of *Lecture Notes in Computer Science*, pp. 109–116, Tübingen, Germany, November 2002.
- [21] J. R. Parker, *Algorithms for Image Processing and Computer Vision*, John Wiley & Sons, New York, NY, USA, 1996.
- [22] R. Poli, “Genetic programming for image analysis,” in *Genetic Programming*, J. R. Koza, D. E. Goldberg, D. B. Fogel, and R. L. Riolo, Eds., pp. 363–368, MIT Press, Cambridge, Mass, USA, 1996.
- [23] S. Roy and I. J. Cox, “A maximum-flow formulation of the N-camera stereo correspondence problem,” in *Proceedings of the 6th International Conference on Computer Vision (ICCV '98)*, pp. 492–502, Bombay, India, January 1998.
- [24] J. Shi and J. Malik, “Normalized cuts and image segmentation,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 22, no. 8, pp. 888–905, 2000.
- [25] M. Sonka, V. Hlavac, and R. Boyle, *Image Processing: Analysis and Machine Vision*, PWS, Pacific Grove, Calif, USA, 2nd edition, 1999.

- [26] E.-G. Talbi, "A taxonomy of hybrid metaheuristics," *Journal of Heuristics*, vol. 8, no. 5, pp. 541–564, 2002.
- [27] R. Urquhart, "Graph theoretical clustering based on limited neighbourhood sets," *Pattern Recognition*, vol. 15, no. 3, pp. 173–187, 1982.
- [28] O. Veksler, "Image segmentation by nested cuts," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR '00)*, vol. 1, pp. 339–344, Hilton Head Island, SC, USA, June 2000.
- [29] S. Voß, "Meta-heuristics: the state of the art," in *Proceedings of the Workshop on Local Search for Planning and Scheduling (ECAI '00)*, A. Nareyek, Ed., vol. 2148 of *Lecture Notes in Computer Science*, pp. 1–23, Springer, Berlin, Germany, August 2001.
- [30] Z. Wu and R. Leahy, "An optimal graph theoretic approach to data clustering: theory and its application to image segmentation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 15, no. 11, pp. 1101–1113, 1993.
- [31] M. Yoshimura and S. Oe, "Evolutionary segmentation of texture image using genetic algorithms towards automatic decision of optimum number of segmentation areas," *Pattern Recognition*, vol. 32, no. 12, pp. 2041–2054, 1999.
- [32] C. T. Zahn, "Graph-theoretic methods for detecting and describing gestalt clusters," *IEEE Transactions on Computers*, vol. 20, no. 1, pp. 68–86, 1971.

Abraham Duarte: Departamento de Ciencias de la Computación, ESCET-Universidad Rey Juan Carlos (URJC), Campus de Móstoles, 28933 Madrid, Spain

Email: abraham.duarte@urjc.es

Ángel Sánchez: Departamento de Ciencias de la Computación, ESCET-Universidad Rey Juan Carlos (URJC), Campus de Móstoles, 28933 Madrid, Spain

Email: angel.sanchez@urjc.es

Felipe Fernández: Departamento de Fotónica, FI-Universidad Politécnica de Madrid (UPM), Campus de Montegancedo, 28860 Madrid, Spain

Email: felipe.fernandez@es.bosch.com

Antonio S. Montemayor: Departamento de Ciencias de la Computación, ESCET-Universidad Rey Juan Carlos (URJC), Campus de Móstoles, 28933 Madrid, Spain

Email: antonio.sanz@urjc.es

14

Model-based image analysis using evolutionary computation

Jean Louchet

Artificial evolution provides powerful techniques in model-based image analysis and model identification. In this chapter, we show how evolution strategies can actually widen the scope of Hough transform generalisations and how some of their variants and extensions, in particular the Parisian approach, can efficiently solve real-time computer vision, sensor fusion, and robotics problems with little reference to more traditional methods.

14.1. Introduction

14.1.1. Image synthesis: a source of models and heuristics for image analysis

From the seventies, image synthesis has been undergoing a huge development with its own subdomains, and obtained results with high visual quality as needed by the image and film industry. In parallel, efforts were made to make these techniques more affordable, using specialised architectures, simulators, and algorithmic research.

On the other hand, while the image synthesis community showed limited interest in analysis, the machine vision community was increasingly using synthesis as a reference in its own work. Synthesis was first used as a tool to assess the performance of image analysis algorithms and systems, but the influence of the “knowledge-based systems” philosophy soon encouraged the image analysis community to use it as a technique to express and manipulate a priori knowledge on the scene to be analysed, rather than merely a tool to build a posteriori evaluation systems. The status of the vision algorithm has been evolving since, to become an engine to instantiate the parameters of a scene model, using pixel calculation: as computational tools have been developing, synthesis evolved from being a conceptual, indirect reference in vision to an operative reference now using common models and algorithms.

In its early hours, image analysis was widely inspired by signal analysis and its success story can be partly explained now through the relevance of contours as probable projections of 3D edges. Other authors introduced region algorithms

whose philosophy is more oriented towards the detection of probable projections of homogeneous facets from the 3D scene, leading to quite different pixel computation methods. While contour algorithms rely on discontinuities (which may have a psychophysic flavour), “region” algorithms rely more explicitly on physics, as a region may be described in its essence as a set of geometric (connectivity) and photometric (material homogeneity) properties of the objects in the scene.

Here, modelling is used to formalise *a priori* knowledge about the scenes. This knowledge can be divided into two components. The first one is specific to the scene studied: it consists essentially in a geometric description, completed if necessary with parameters linked to rendering texture and photometric attributes (such as albedo, diffusion diagrams, light sources). The second one is general knowledge about physics: general optics, rules about objects collisions (if applicable), and so forth.

14.1.2. Heuristics and segmentation

Among all other segmentation approaches (contours, interest points, etc.), image segmentation into regions is probably the most natural (if not always the easiest to implement). Let us examine why.

In a 3D scene, an elementary facet (or, more generally, a portion of a surface) is the only simple entity which actually supports a physical process (here, the reflection and diffusion of light) involved in the formation of a natural image.¹ Attempts to accurately model what is happening when light interacts with an object will naturally lead to introduce colour (wavelength dependent) and brightness values, complex reflection and diffusion models (the optician’s point of view), and texture models (the probabilistic point of view) as parameters attached to the physical surfaces of objects. The same would obviously be untrue with other primitives as edges or corners. Thus, all the radiometric properties of usual objects are in fact properties of their surfaces. These 3D real-world properties of objects in the scene may hopefully be translated into properties of the corresponding zones in the images.

This is why region segmentation is the most natural approach to segmentation. Most *a priori* knowledge about the scene—at least the general laws of optics—can be expressed as physical properties. In spite of all the instrumental artefacts that can appear when the digital image has been created (geometrical distortion, optical aberrations, sampling errors, noise, etc.), these physical properties will have to be translated into physical properties at the image level.

If, as it often happens, we have no specific knowledge on the scene, it is then reasonable to suppose that two points in the scene, if they belong to the same object, will be more likely made from the same material than if they belong to two different objects. Now, if we project this assertion onto the image plane, the

¹This may not be the case, for example, in radar images where diffraction is the dominant physical process involved.

two corresponding pixels will probably look more closely like each other than if they belonged to two different objects. This pixel similarity may be evaluated on a photometric level (having the same colour), on a geometrical/statistical level (having the same texture), or even sometimes on more elaborate levels like having the same apparent velocity, having the same type of behaviour or, for greater numbers of points, having apparent velocities that can be translated into a consistent model of a solid object in motion.

This is where the “homogeneity predicate” has been introduced. A portion of an image will be said to satisfy the homogeneity predicate if all (or most) of its pixels present a sufficient level of similarity as defined above. The choice of a homogeneity predicate is essential, because it is the formal expression of all the knowledge we have on the image in order to understand it as containing 2D projections of significant 3D real-world objects.

The problem now is to find which is the best way to divide the image into regions so that a homogeneity predicate is fulfilled as nicely as possible by each region. As always, there will be a very large number of resolution techniques: some of them will normally give an excellent solution but will be expensive to program and/or to run, other ones will never give as good solutions as above but will be cheap and fast to run, other ones will give rough results in a very short time and then progressively refine them (this is the kind of things roboticians love), and so on.

To summarise, writing a region segmentation algorithm should ideally consist of two parts. The first part consists of listing the properties one would expect from the result of segmentation, this is done by translating into the image plane the general physical properties we already know about the scene: this is “problem-specific knowledge.” The second part consists of writing or readapting an algorithm able to solve the problem, this part of the task involves “general knowledge on problems resolution.” This is but the very basic ideas underlying the theory of artificial intelligence and knowledge-based systems.

For example, with a given homogeneity predicate, we know that it is possible to segment the same image using very different techniques: split, merge, split and merge, region growing, and so forth. These techniques will have different cost and performance, but will not give essentially different results. Conversely, if with one application involving one style of images (say, detecting elks in a forest using a colour camera) we have found a very good combination of a homogeneity predicate and a resolution engine, then when facing a different vision problem (say, detecting an aircraft in the sky using an infrared camera), we will probably have to write a new homogeneity predicate, but may probably reuse the same resolution engine. In an ideal world, any predicate may be combined with any resolution engine.

Of course, practice is not quite as clean, and many resolution engines are very much specific to one homogeneity criterion. Whatsoever, it is still advisable to separate as much as possible the image-specific or scene-specific knowledge from the general (possibly mathematical) knowledge on resolution techniques. This will become truly obvious when using classical optimisation methods to solve image

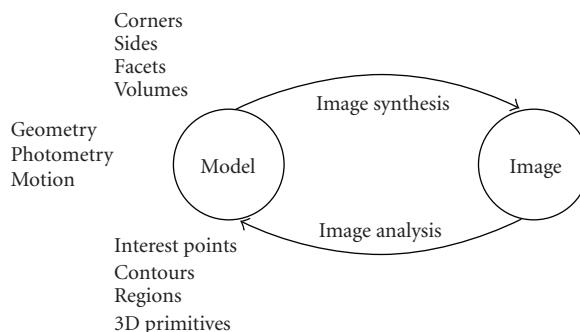


FIGURE 14.1. 3D model and segmentation.

processing problems that have first been translated into a cost function to be optimised (e.g., contour tracking, snakes, or the apparent motion constraint equation for motion analysis, etc.).

The role of image analysis is to use image data in order to build a “cognitive” model of the scene, similar to the scene description models used as input data by image synthesis algorithms. Image analysis may thus be considered as the inverse process of image synthesis and scene models (viewed as image synthesis input or as image analysis output) should ideally use the same description language (Figure 14.1).

One of the first questions arising is the choice of a scene modelling language. A majority of modelling languages use facets and polyhedra, which is well adapted to the usual environments containing opaque objects with geometrical regularity, but other primitives and representations may be more suitable in other instances. In general, image segmentation will produce polyhedral structures: corners, edges, facets, and their photometric attributes, grouped into objects. This may be completed with a description of light sources, velocities and deformation, object interaction, and individual or collective behaviour.

The first task of an image segmentation algorithm will then be to build such a polyhedral object database. A most naive way to solve this problem would be to exhaustively explore the space of all possible scene models, or just of all “reasonable” (i.e., polyhedral) scene models. To this end, one should find a fast and efficient image synthesis algorithm, feed it with all these models, and compare the output image with the reference image to be analysed in order to decide which model predicts it the best.

The principle above is not quite as unrealistic as it would appear at first glance. It has been successfully used in the early seventies to determine aircraft 3D attitude from trajectography image sequences. As the object is centred and its shape is known, there are only 3 parameters to identify, making the search space small enough to explore it fast without using lots of refinement. In more complex cases, the same principles may remain valid, though requiring smarter parameter space exploration techniques as we will see through the examples given further.

Whatever exploration method has been chosen, one has to optimise the similarity of a synthesized image with a real-world image. How is it possible to measure the degree of similarity between two images? Mean square error methods are easily implemented but seldom satisfactory as too sensitive, for example, to small translations. This is why similarity functions should be based on image attributes as little sensitive as possible to the image transforms to which we want the algorithm to be tolerant. In practice, the image analysis problem is expressed as the minimisation of a cost function which will be chosen according both to semantic and to computational criteria.

However, in a vast majority of applications, while this approach of image analysis (as the inverse problem of image synthesis) may help to specify the problem, it will not necessarily give a resolution method for free. This is precisely where the history of image processing can be reconsidered (or rewritten) under a new angle of view.

Pioneering work in image processing may be viewed as implicit, unexpressed attempts to dramatically reduce the huge combinatorics found when trying to solve image analysis problems as inverse problems of image synthesis. To make this problem tractable, one had to exploit as much as possible general knowledge about the scene (or the class of scenes to be analysed) and create heuristics to reduce exploration time. This will be made clearer through an example.

General knowledge about scenes containing man-made objects tells that many objects have polyhedral shapes and are made from homogeneous materials. Knowledge about geometry and optics tells that 3D facets will be projected on the image as polygons, or at least regions which are probably connex. Knowledge on optics (and common sense) tells that a facet from an object made from a single material will keep some homogeneity properties in its image projection (under usual lighting conditions). Similarly, 3D edges will project as straight lines, and as the two facets adjacent to the 3D edge will (if visible) be illuminated under different angles, there will be some sort of contrast between the two sides of the line. To exploit these properties and explore the search space more efficiently, it was then obvious that it is advantageous to explore first the scene models containing facets that project onto homogeneous regions, or containing edges that project onto highly contrasted lines. This is at least one interpretation of how region-based and contour-based segmentations have come to life. More generally, to each 3D model entity (corner, edge, facet, etc.), it is possible to associate a (hopefully local) property in the image: modelling these properties and detecting image entities which fulfil them are the heart of image processing.

The way the geometrical primitives in a polyhedral model are hierarchised can be translated into a hierarchy of image processing entities [5]:

- (i) an interest point, as a pixel likely to be the image projection of a corner of a (at least locally) polyhedral object;
- (ii) a contour, as a curve or an image line likely to be the projection of a side or a boundary of an object;
- (iii) a region, as a connex subset of the image likely to be the projection of a fragment of an object or facet surface;

TABLE 14.1. Correspondence between image analysis and synthesis primitives.

Dimension	Segmentation type	Synthesis equivalent	Technique	Property
0	Interest point	Corner	Local extremum	Concentration
1	Contour	Side	Gradient	Disparity
2	Region	Facet	Colour, texture	Uniformity
3	Dynamics	Motion, stereo	Correspondence	Similarity, disparity length

- (iv) analysing motion in an image sequence, as an attempt to interpret it as a three-dimensional kinematic or, if possible, dynamical or behavioural model.

It is therefore possible to build a correspondence table between the primitives used in image synthesis and the main image segmentation methods (see Table 14.1).

It follows that for a given class of scenes, any segmentation technique is just as relevant as the corresponding description language. Detecting regions, contours, and interest points would be meaningless if these primitives did not appear as primitives in the scene modelling technique chosen.

14.2. Artificial evolution seen as a heuristic for parameter space exploration

14.2.1. Introduction

As discussed earlier, many segmentation techniques have been motivated by the difficulty to optimise a geometrical model of the scene using conventional techniques. However, the best-known attempt to bypass segmentation is Hough's transform [7], which can be viewed as a simple technique to explore a parameter space. It consists of using a vote technique to map the parameter space into the set of positive integers. To this end, each feature locally detected in the image votes for all the points in the parameter space able to give birth to the given feature. Once each feature has voted, the parameter set getting the highest number of votes is elected: it is considered as the best possible model parameterisation, as it is able to explain a greater number of image features than any other model.

Unfortunately, the Hough transform and its extensions [15] suffer from a computational complexity which increases with the complexity of patterns in the parameter space and the number of unknowns. In spite of many clever heuristics, the Hough transform becomes hardly usable if the dimension of the search space is over 3 or 4.

14.2.2. Direct application: the Hough transform revisited by artificial evolution

Hough's technique leads to calculate vote values over the whole parameter space then explore it exhaustively in order to detect the best solutions. An appealing



FIGURE 14.2. Result of the classical Hough transform (image 288×352).

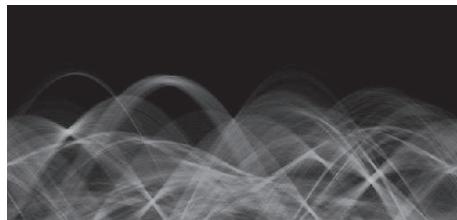


FIGURE 14.3. Parameter space (θ, ρ) containing the Hough accumulator of the preceding image (image 256×300).

alternative is to directly explore the search space [14, 17, 19]. Evolution strategies [1, 16] give an interesting opportunity to only calculate values in the part of the parameter space where population individuals actually are, rather than on the whole parameter space.

It is thus possible to build an “evolutionary version” of the Hough transform:

- (i) the population is a set of points in the parameter space,
- (ii) the fitness function measures the match quality of the individual to be evaluated, with the features of the given image,
- (iii) selection: tournament,
- (iv) mutation: Gaussian noise.

In the classical context of the Hough transform, detecting straight lines in images, there is no significant advantage towards using the original or the evolutionary version. The classical method, where each point (x, y) in the image votes for the set of the (θ, ρ) in the parameter space such that $\rho = x \cos \theta + y \sin \theta$, is reasonably fast and memory consuming (Figures 14.2 and 14.3); the evolutionary version which—thanks to a sharing operator—enables to detect several solutions (Figure 14.4), does not have an easy-to-define execution time, which makes their performance difficult to compare.

More interestingly, when dealing with greater numbers of parameters, exploring and saving the parameter space become prohibitive, while the evolutionary version remains reasonable in terms of memory and computing effort.



FIGURE 14.4. Result of the evolutionary Hough transform.

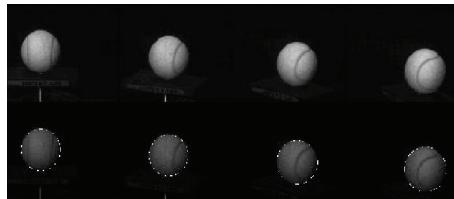


FIGURE 14.5. Four original images from the “tennis ball” sequence (top) and the results of circle tracking using an evolutionary Hough transform (bottom).

The next two examples are taken from a student project of A. Eckman in KTH.² The first one (Figure 14.5) consists of detecting circles with unknown diameters (a moving ball) in an image sequence. The individuals are triplets (a, b, r) containing the parameters of the circles

$$(x - a)^2 + (y - b)^2 = r^2. \quad (14.1)$$

An individual’s fitness is defined as the average gradient norm taken on 40 points randomly chosen on the circle. The algorithm parameters are summarized in Table 14.2.

It is interesting to point out a property of the evolutionary Hough transform. If the object’s motion is small enough between two consecutive frames, the algorithm will track the object’s movement, unlike the deterministic version which has to begin calculations from scratch at every new frame whatever the degree of consistency between consecutive images. In other terms, while the Hough transform has never been seen since fast and evolutionary algorithms are generally regarded as slow, the evolutionary Hough transform gets true real-time properties.

This simple version of the algorithm has been written in C without any specialised library.

²The Royal Institute of Technology, Stockholm, Sweden.

TABLE 14.2. Parameters of the circles detection algorithm.

Population size	100
Selection	2 - tournament
Mutation rate (%)	15
Mutation amplitude r	10
Mutation amplitude a, b	40
Crossover rate (barycentric) (%)	5
Generations per frame	800 to initialise, then 240 per frame



FIGURE 14.6. Image of galaxy AM 0644-741 taken by the Hubble telescope (left) and result of the evolutionary Hough ellipse detector (right).

The second example (Figure 14.6) deals with recognising an ellipse in an image, using the same algorithm and parameters but with a larger genome (a, b, r_x, r_y, α) corresponding to the ellipse equation

$$\begin{aligned} x &= a + r_x \cos \alpha \cos \theta + r_y \sin \alpha \sin \theta, \\ y &= b - r_x \cos \alpha \sin \theta + r_y \sin \alpha \cos \theta. \end{aligned} \quad (14.2)$$

14.2.3. Camera calibration and attitude determination

Another application of the generalised evolutionary Hough transform is detecting the 3D position of a flying helicopter in order to assist its landing on a ship. A camera pair (video or infrared) is used in order to determine the 6 position and angle parameters of the helicopter, whose shape is known. The fitness function is based on a comparison between the silhouette A extracted from the stereo images and the silhouette B predicted by image synthesis:

$$\text{fitness} = \frac{\text{area}(A \cap B)}{\sqrt{\text{area}(A)\text{area}(B)}}. \quad (14.3)$$

We tested the algorithm first on realistic scenes (obtained using a Revell scale model), then in simulation (using noisy synthetic images) to be able to compare with reliable ground truth data. While limiting the algorithm to 30 generations with a population of 60 individuals, which ensures video rate processing with

a cheap laptop computer, the average errors taken on 29 trials are

- (i) average position error (using the distance between cameras as a unit):
 - (a) x and y , 0.01,
 - (b) z (depth), 0.04;
- (ii) average error on angles (in degrees):
 - (a) α (rotation of the helicopter around the vertical axis) 0.86,
 - (b) β (pitch) 1.74,
 - (c) γ (roll) 2.14.

The low precision on γ is certainly due to the low variations of the silhouette when the helicopter rotates around its main axis.

14.3. Collective representation: the Parisian evolution and the fly algorithm

In most classical approaches of artificial evolution, as illustrated above, each individual represents a potential solution of the problem to be solved. The best individual is retained as the solution. This philosophy is well adapted, for example, to detect predetermined patterns in images, provided that each object to be detected correspond to a unique point in the parameter space.

However, in many applications, it is not possible to interpret the image or the image sequence just as the sum of separable image entities, each one corresponding to a particular point to be discovered in the parameter space of a suitable model. It is often more convenient to describe the image as the combination of the outputs of a model which has been applied to a large subset of a certain parameter space. In the following sections, we will examine how it is possible to exploit the “Parisian” (or “individual”) approach [3] which considers that the solution will not be represented by a single individual in the population but by the whole population (or at least by an important fraction of it).

14.3.1. The principles

Many applications of stereovision to robotics do not require an exhaustive geometrical description of the scene, as it would be given by costly classical stereovision methods based on image segmentation. Vote-based (Hough-inspired) methods are generally unpractical because of the high dimensionality of the parameter space. The method presented here is an example of the third alternative, evolving a population of 3D points so that they will concentrate onto the visible surfaces of the objects in the scene. This bias towards simplicity³ will be exploited even into the way the fitness function is written—which is a major contribution to the total calculation load. We will then show that extending this “fly algorithm” to time sequences opens the way to real-time processing even with low-cost hardware.

In the fly algorithm, the population of flies evolves in order to tend to spread onto the visible surfaces in the scene. The fitness function is designed in order to

³As such, this is in a way similar to the work of Eberhart and Kennedy [4] on particle swarms, another optimisation tool inspired by artificial life concepts.

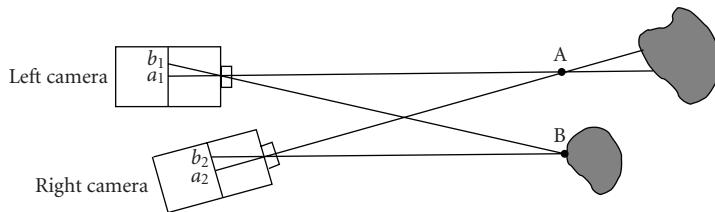


FIGURE 14.7. Pixels b_1 and b_2 , projections of fly B , have identical grey levels, while pixels a_1 and a_2 , projections of fly A , which receive their illumination from two different physical points on the object's surface, have different grey levels.

give higher fitness values to flies whose projections into the different cameras are consistent. In other terms, if a fly is not on an object's surface, then its calculated projection in one image will be in reality the projection of the first real object on the line from the lens to the fly: there is no reason why the fly's projections would be similar or correlated in any way (Figure 14.7).⁴ Conversely, if the fly is on an object's surface, then its projections into any camera will actually be the projection of the corresponding point of the real object, and will usually be similar (same colour, same texture, high correlation, etc.). The fitness function translates the degree of similarity of the fly's projections onto the different cameras used (usually two).

An individual (a fly) is defined as a point in space, whose chromosome is its set of coordinates (x, y, z) . The coordinates of the fly's projections are (x_L, y_L) in the image given by the left camera and (x_R, y_R) in the right camera. The calibration parameters of the cameras are known, and therefore x_L, y_L, x_R, y_R may be readily calculated from x, y, z using projective geometry [6]:

$$\begin{pmatrix} x_L \\ y_L \\ 1 \end{pmatrix} \equiv F_L \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} \begin{pmatrix} x_R \\ y_R \\ 1 \end{pmatrix} \equiv F_R \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix}, \quad (14.4)$$

where F_L and F_R are the projective matrices (3, 4) of the left and right cameras.

The fitness function exploits this property and evaluates the degree of similarity of the pixel neighbourhoods of the projections of the fly onto each image, giving highest fitness values for the flies lying on objects surfaces:

$$\text{fitness} = \frac{G}{\sum_{\text{colours}} \sum_{(i,j) \in N} (L(x_L + i, y_L + j) - R(x_R + i, y_R + j))^2}. \quad (14.5)$$

⁴This may not be true if the surfaces do not follow Lambert's law, which assumes that for a given illumination, the object's radiance does not depend on the observer's position. Most surface-based stereovision methods are also sensitive to this property.

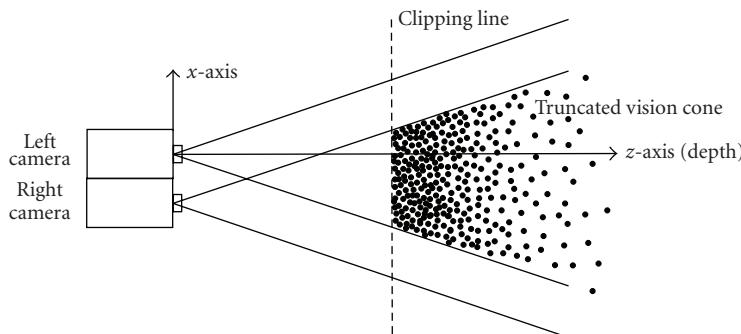


FIGURE 14.8. The fly population is initialised inside the intersection of the cameras 3D fields of view.

- (i) (x_L, y_L) and (x_R, y_R) are the coordinates of the left and right projections of the current individual (see Figure 14.7).
- (ii) $L(x_L + i, y_L + j)$ is the grey value of the left image at pixel $(x_L + i, y_L + j)$, similarly with R for the right image.
- (iii) N is a neighbourhood introduced to obtain a more discriminant comparison of the fly's projections.

In colour images, square differences are calculated on each colour channel.

The numerator G is a normalizing factor designed to reduce the fitness of the flies which project onto uniform regions. It is based on an image gradient norm calculation. The best experimental results are obtained when G is defined as the square root of Sobel's gradient norm: highest fitness values are obtained for flies whose projections have similar and significant pixel surroundings. G has been empirically defined as

$$G = \sqrt{\sum_{(i,j) \in N} (L(x_L + i, y_L + j) - L(x_L, y_L))^2}. \quad (14.6)$$

The fitness function is corrected, using a local averaging of grey levels, in order to eliminate the constant component of the images and reduce the fitness function's sensitivity to unbalanced brightness or sensitivity adjustments in the cameras.

Now, the fitness function contains all pixel calculations. Let us examine the evolution operators.

The population is initialised randomly inside the intersection of the cameras' fields of view (Figure 14.8). The statistical distribution is chosen in order to obtain uniformly distributed projections in the left image. The values of z^{-1} are uniformly distributed between zero (or $1/d_{\max}$) and $1/d_{\min}$. Thus, the density of flies decreases gently with depth.

Selection uses the tournament technique.

2D sharing is based on the densities of the 2D projections of flies on the images. It reduces the fitness values of flies projecting into crowded areas to prevent

them from getting concentrated into a small number of maxima [2]. It reduces each fly's fitness by $K \times N$, where K is a “sharing coefficient” and N the number of flies which project into the left image within a distance R (“sharing radius”) from the current fly, given by

$$R \approx \frac{1}{2} \left(\sqrt{\frac{N_{\text{pixels}}}{N_{\text{flies}}}} - 1 \right). \quad (14.7)$$

Mutation allows extensive exploration of the search space. It uses an approximation of a Gaussian random noise added to the flies' chromosome parameters (x, y, z) . We chose standard deviations $\sigma_x, \sigma_y, \sigma_z$ equal to R , so that they are of the same order of magnitude as the mean distance between neighbouring flies.

Two *barycentric crossover operators* have been introduced, in order to take into account the frequent straight lines and planar surfaces existing in real-world scenes. The first one builds an offspring randomly located on the line segment between its parents: the offspring of the two flies $F_1(x_1, y_1, z_1)$ and $F_2(x_2, y_2, z_2)$ is the fly $F(x, y, z)$ defined by $\overrightarrow{OF} = \lambda \overrightarrow{OF_1} + (1 - \lambda) \overrightarrow{OF_2}$. The weight λ is chosen using a uniform random law in the interval $[0, 1]$ or $[-0.5, 1.5]$.⁵ Similarly, the second crossover operator uses three parents and determines the offspring F such that $\overrightarrow{OF} = \lambda \overrightarrow{OF_1} + \mu \overrightarrow{OF_2} + (1 - \lambda - \mu) \overrightarrow{OF_3}$ in the parents plane, using two random weights λ and μ .

14.3.2. Real-world images: processing stereo sequences

Results of the algorithm on static indoor scenes have been published in [12]. There are several possible approaches to extend the static fly algorithm to stereo image sequences. The simplest is the *random approach*, which consists of keeping the same population evolving through frame changes. Thus, only the images (and therefore the parameters used by the fitness function) are modified while the fly population evolves. When motion is slow enough, using the results of the last step speeds up convergence significantly compared to using a totally new random population. Here, the population of flies is used as a memory of space, allowing the algorithm to exploit the similarity between consecutive scenes rather than forgetting the information collected during the previous steps. This does not require significant algorithm changes. To improve detection of new objects appearing in the field of view, we introduced an extra mutation operator, *immigration*, which permanently creates random new flies in a way similar to the procedure already used to first initialise the population: used with a low probability (1% to 5%), this favours a convenient exploration of the whole search space.

The dynamic approaches introduce explicit velocity components into each fly's genome. The advantage is a better precision for a given number of generations

⁵It is generally accepted that a barycentric crossover operator with positive coefficients has contractive properties which may be avoided by extracting weights from a larger interval. Here, the choice depends on whether it is desirable or not, to fill in surfaces whose contours have already been detected, rather than to extend them.



FIGURE 14.9. Real-time processing on a highway. The flies (black dots) concentrate on contrasted road edges and other cars.



FIGURE 14.10. Alarm values (car on the highway).

or evaluations, but this goes with a significantly higher calculation cost at each generation, and therefore a smaller number of generations in a given time interval. The best trade-off will depend on the scene style and complexity of motion.

In what follows, as we consider a vision system embedded into a mobile robot, we will use the simple random approach described above, but update the flies' positions at each generation, using the information available about the robot's motion in order to give better initial conditions and allow faster convergence of the fly population. This will be the basis of the proprioceptive fusion described in Section 14.3.4.2.

To process faster motion, it is possible to extend the chromosome by enriching it with velocity components: each fly is now represented by the 6-uple $(x, y, z, \dot{x}, \dot{y}, \dot{z})$ to keep its own velocity information, in a way reminiscent of a Markov process. These extensions are described in more detail in [12]. The following examples (Figures 14.9–14.12) show the results of the fly algorithm in real time in a car equipped with two cameras.⁶ An “alarm value” is given to each fly, depending on its fitness and proximity to the anticipated car trajectory. An emergency braking system uses the sum of alarm values as an input.

14.3.3. Application to robotics: fly-based robot control

Scene and obstacle representation by a population of flies is the most unusual way to represent the physical environment of a mobile robot. Classical robot controllers

⁶Data captured and processed by O. Pauplin in the framework of a cooperative project between the IMARA and COMPLEX teams at INRIA, Rocquencourt, France.



FIGURE 14.11. Pedestrian.

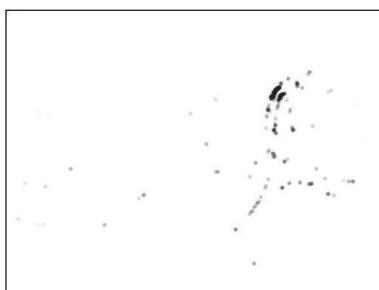


FIGURE 14.12. Alarm values (pedestrian).

are not designed to use flies as an input. Of course, one could think the natural way to integrate the fly algorithm into a mobile robot would be to interface it with existing controllers by building a translator, able to transform the fly-based scene representation into a more classical representation, for example, a polyhedral model, which a classical controller would be able to use.

Our position is that such a translator would almost certainly remove most of the advantages of the fly algorithm, in particular in terms of speed and simplicity, and that it is wiser to develop new navigation methods—or adapt existing ones to the fly input. The results shown in this section (Figures 14.13–14.19) have been obtained by A. Boumaza using his simulator which simulates all the perception-action loop:

- (i) a stereo camera pair simulator (image synthesis),
- (ii) the fly algorithm,
- (iii) the trajectory planning algorithm,
- (iv) a kinematic simulator of the robot's motion.

A. Boumaza first developed a trajectory planner based on classical potential-based methods. A force derived from the addition of an attractive (target) and a repulsive (flies) potential was acting as a steering command: the blockage situations were resolved using two heuristics creating a new force attracting the robot out of the potential minimum (random walk- and wall-following methods).

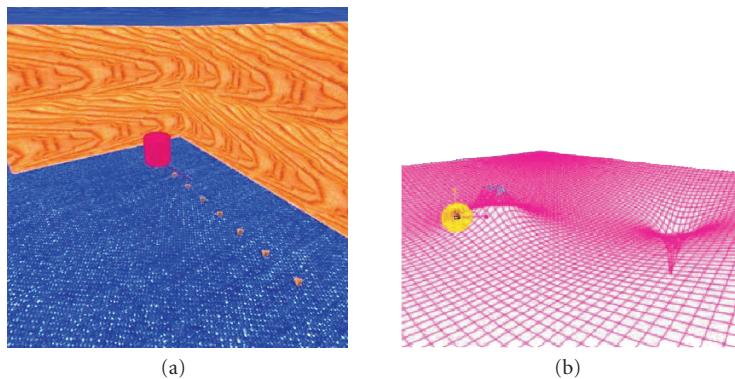


FIGURE 14.13. The robot facing an obstacle (a) and the corresponding harmonic function (b).

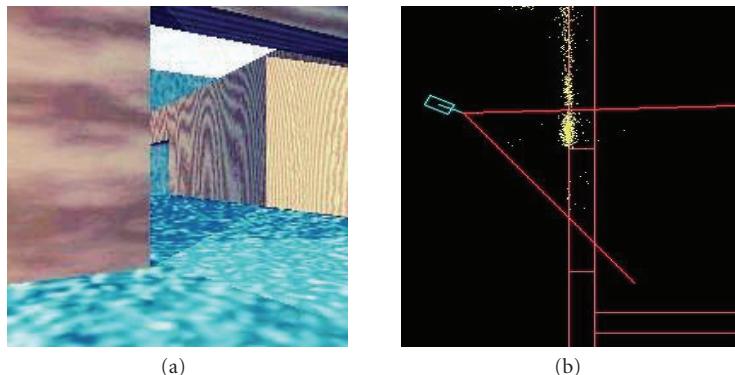


FIGURE 14.14. A synthetic image of the scene as seen by one of the robot's cameras. (b) shows the flies along the door, as previously detected by the robot.

Not surprisingly, the robot encounters blocking situations, where the control force oscillates around zero (potential local minima). To resolve this, we implemented two heuristics [9, 21] to create a new attraction force out of the local minimum.

With the random walk method, the algorithm creates random secondary targets and uses in priority those having the smaller number of obstacles between themselves and the robot, and between themselves and the target. This secondary target replaces the main target as long as necessary [2].

The wall-following method just modulates the direction of the attraction vector, so that the new resulting force becomes roughly parallel to the obstacle's orientation.

The next more efficient potential field-based path planner that we developed is based on harmonic functions [9]. A harmonic function is a function which

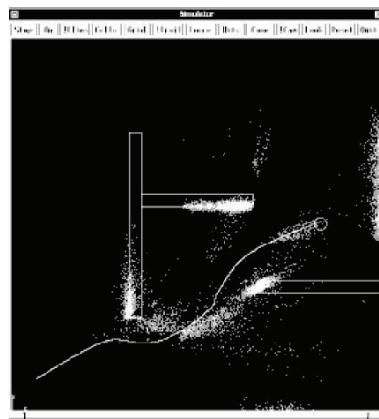


FIGURE 14.15. A direct trajectory without blockage situations.

satisfies Laplace's equation

$$\Delta U = \frac{\partial^2 U}{\partial x^2} + \frac{\partial^2 U}{\partial y^2} = 0. \quad (14.8)$$

The robot's vision system uses the population of flies to build its own representation of the environment as the samples of a harmonic function. Values of the function at the target and obstacle positions are modelled as Dirichlet boundaries: 1 for obstacles, 0 for the target position. The harmonic function is built iteratively using a finite difference operator, such as the Gauss-Seidel operator which replaces the value of a point with the average value of its four neighbours. After convergence, we end up with a smooth function having a single minimum at the target position.

One of the interesting properties of harmonic functions is the absence of local minima, which in our case eliminates the blockage situations found in the first simulator.

The steering command of the robot is the gradient of the harmonic function. Linear interpolation is used when the robot position falls between grid points. During the movement towards the target, new obstacles detected by the fly algorithm are introduced into the harmonic function according to the local fly density, as high potential Dirichlet boundaries. Since the harmonic function is constantly iterated, there will be a constantly updated obstacle avoiding path for the robot to follow (Figure 14.17). The Dirichlet boundaries are built using high-fitness flies in the robot's field of view.

14.3.4. Sensor fusion

Sensor fusion plays a central role in a robot's perception system. Many classical approaches are based on Bayes' theorem. This is not possible here as Bayes is only

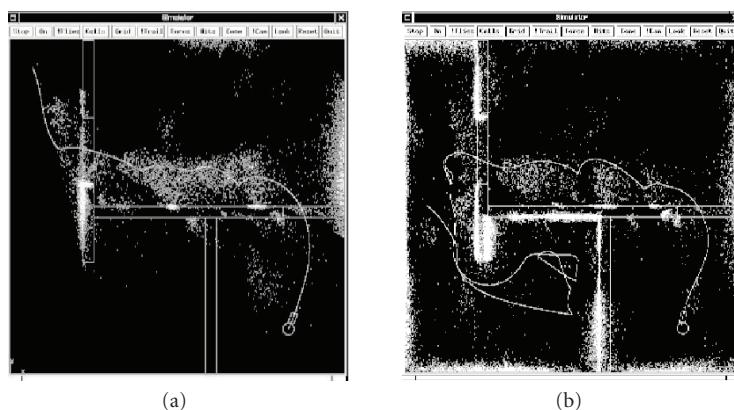


FIGURE 14.16. Two examples of obstacle avoidance using secondary targets. The circle represents the main target.

applicable in probabilistic approaches, and the results of the fly algorithm cannot be formally considered as probabilities. However, it is possible to introduce an efficient exteroceptive and proprioceptive sensor fusion, by tickering into the heart of the fly algorithm.

14.3.4.1. Exteroceptive sensor fusion: a multisensor fitness function

The new fitness function should integrate information issued by all the exteroceptive sensors. As stated above, it is difficult to give a formal mathematical justification of how to extend the expression of the fitness function to integrate several sensors. In qualitative terms, if a fly's position is in accordance with several independent sensors, its fitness must be increased accordingly. Conversely, a fly confirmed by the visual sensor should still be considered seriously as a real obstacle even if unnoticed by another sensor (e.g., a visible obstacle covered with sound damping material). We defined each sensor's contribution to the fitness function as follows: if a fly is located inside the vision angle of a triggered detector, and its distance to the detector matches the obstacle distance given by the detector accurately enough, then the fly's fitness is increased by a given percentage B .

If the fly's position does not match any rangefinder information, then the fitness remains unchanged. We chose a multiplicative rather than an additive contribution to the fitness because of the low angular resolution of the ultrasonic sensors: this prevents other flies at the correct distance inside the field of an ultrasonic sensor from getting high-fitness values even if they are inconsistent with image data.

In addition to the integration of additional sensors into the fitness function, we introduced an additional *immigration* operator: new flies are introduced into the population with a bias in favour of 3D positions given by the sonars. This helps detecting close obstacles which might otherwise have been overlooked by the system.

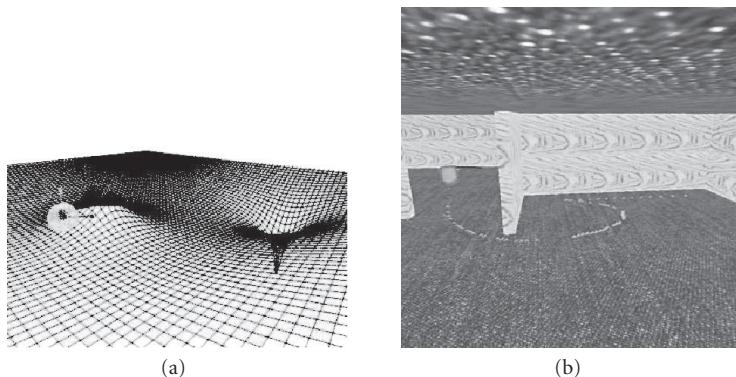


FIGURE 14.17. A harmonic function used for obstacle avoidance and the resulting robot trajectory.

This allows the fusion of an arbitrary number of sonars (or any similar exteroceptive sensors) at the fitness level, but it is not appropriate to the integration of proprioceptive sensor information, which we examine in the following sections.

14.3.4.2. Proprioceptive sensor fusion: the proprioception genetic operator

Proprioceptive sensors provide information about the robot's own state, in particular its position. Inertial sensors and odometric sensors (using wheel rotation coders) are among the most commonly used sensors to estimate the robot's position. Without such sensors, the fly algorithm is able to optimise in a quasicontinuous way the population of flies, which will follow the scene's motion while the robot is moving slowly. However, information about the robot's motion helps to update the positions of flies and speeds up convergence, especially in case of fast robot movements.

In most applications, wheel rotation is under control of the trajectory planner. The actual robot trajectory does not exactly match the target trajectory due to "trajectory noise" caused by factors such as wheel slipping, tyre wear, or rough ground surface. In our simulator, the robot's actual position is simulated by adding a Gaussian noise to the planner's command, but the sensor fusion algorithm only gets what it would get in the real world, for example, the odometric estimation which is strictly identical to raw trajectory planner data. Integration of odometric information is performed through updating the flies' 3D coordinates in accordance with the motion of the robot's coordinate system. Experiments show that flies' convergence, which otherwise needed around 10 generations, is now satisfactory after less than 3 generations in spite of the low precision of odometric information.

In this example (Figures 14.18 and 14.19), the robot is rotating (1 degree per frame and one generation per frame). In Figure 14.18, obtained without proprioceptive fusion, there is some delay in detecting the short range obstacle as the

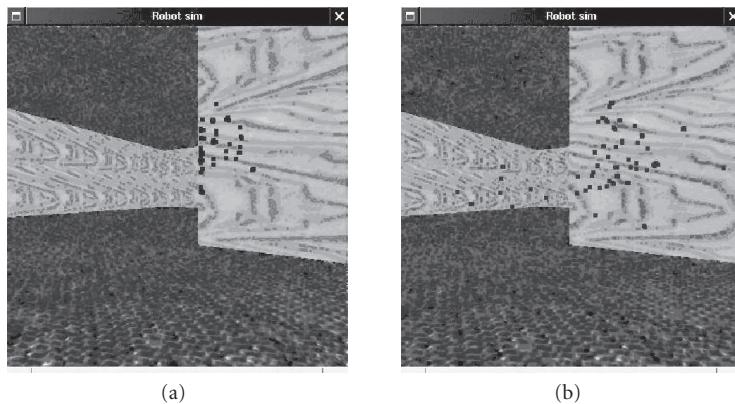


FIGURE 14.18. Frames N and $N + 5$ without proprioception.

flies tend to remain on the further wall, already detected on the preceding images. Figure 14.19 shows the results on the same sequence, with proprioceptive fusion. Updating the flies' positions allows for faster convergence and better detection of the short range obstacle.

14.3.5. Real-time and artificial evolution

Artificial evolution does not have a good reputation in terms of speed. The general reasons for this reputation could be discussed interestingly, but let us examine what happens with the fly algorithm in this respect.

The speed of an evolution strategy strongly depends on the complexity of fitness calculation—which is in the lightweight side with the flies. Anyway, speed in itself is not the ultimate criterion of real time. Real time refers to the ability to exploit the flow of input data so that the system is able to react fast enough for the end user.

Adaptation is probably an important feature here. Generally speaking, evolution strategies are able to cope with a fitness function changing with time during the program's execution [18], which most other optimisation methods cannot do. An autonomous robot must continuously optimise its strategy while the environment is changing. This makes evolutionary methods specially interesting in autonomous robotics applications.

Unlike in image segmentation-based algorithms, image pixels only need to be read when the fitness of a fly has to be evaluated; therefore, the random pixel access allowed by CMOS imagers can be fully exploited and new events in the scene can be processed without the usual frame refreshing delays. With the fly algorithm, each individual evaluation only needs pixel values on a small neighbourhood.⁷

⁷Most commercial CMOS imagers directly deliver small (e.g., 8×8) pixel neighbourhoods on request.

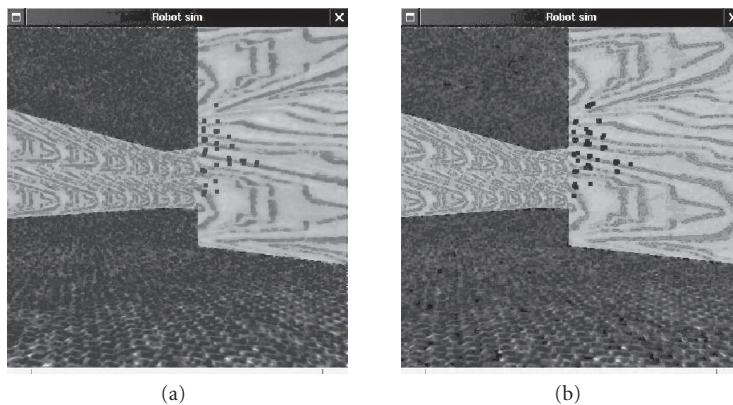


FIGURE 14.19. The same images with proprioception, better detection of obstacles.

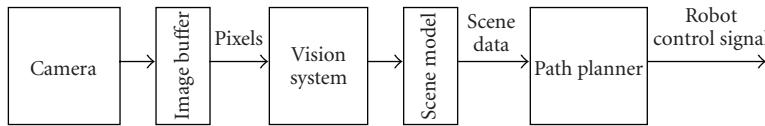


FIGURE 14.20. A classical (synchronous) robot vision-planning system. Defining T as the frame rate period, a classical vision system will use input data which may react to real-world events with a delay up to $2T$, and a synchronous planner must wait until complete update of the vision's output to begin its own work. This leads to a minimal delay of $3T$ in addition to processing time.

CMOS camera technology is well adapted to this requirement as it is able to deliver these values at any time, while conventional image processing techniques do not exploit this feature. This shows that the fly algorithm annihilates the usual delay in input data found in classical image processing algorithms (Figure 14.20).

Similarly, on the output side, the fly algorithm delivers quasicontinuously updated information at any time (Table 14.3). This gives a second similar speed advantage.

Third, if the planner is able to use immediately the quasicontinuous fly algorithm's output, this gives the planner truly faster reaction to scene events. The harmonic function-based planner continuously updates its grid and always uses the freshest information provided by the flies. This conforms to the “anytime algorithms” concept.

To summarise, using artificial evolution in real-time image processing allows one to get free from the usual delays of synchronous processing. Here, the image processing module does not require any image filtering or segmentation. The navigation and control system is not based on an evolutionary algorithm but it keeps the same “anytime” properties which allow it to fully exploit the speed of evolutionary image processing. We can try to list the main advantages:

- (i) fast execution, free from the classical delays of frame delivery found in synchronous vision systems;

TABLE 14.3. Comparison of the fly algorithm with conventional methods.

	CCD sensor + standard approach	CMOS + flies
Image sensor	Delay between capture and restitution	Asynchronous data reading
Image processing (input)	Image segmentation must wait end of capture cycle	Random access to pixels needed by current fitness calculation
Image processing (output)	Not available until segmentation process has ended	Current state of representation always available
Trajectory planner	Must wait for image processing output	Saves 2 acquisition cycles

- (ii) easy programming (essentially editing the fitness function) inside a stable algorithmic architecture;
- (iii) the process will self-adapt to the apparent velocities of objects in the images;
- (iv) ability to exploit and fuse data from other proprioceptive (odometry, inertial) and exteroceptive (radar, acoustic) sensors;
- (v) optimal exploitation of the asynchronous and local properties of state-of-the-art CMOS imagers.

Moreover, most image processing systems are designed through assembling standard operators into an application-specific functional structure. Here, the algorithm's architecture is application-independent, as most of the specific knowledge is coded into the fitness function. Little has to be modified if the program has to be reused into another image processing application.

14.4. Conclusion

In the first part of this chapter, we gave a few examples of how it is possible to extend the Hough transform to an evolutionary version able to solve multiple model-based image analysis problems. The parameter space is explored by an evolving population where each individual is testing a pixel-level predicate. In the second part, we demonstrated through the example of the fly algorithm in stereovision how it is possible to use the Parisian approach of evolutionary computing to build efficient real-time image processing algorithms with interesting asynchronous properties. Unlike classical approaches to stereovision, no image segmentation is required and the results get more complete and accurate with time, which is a valuable property in robotic applications as the balance between speed and accuracy can be adjusted in real time.

The main benefits of evolutionary approaches in model-based image analysis are

- (i) fast processing: processing time depends on population size rather than image size;

- (ii) progressive accumulation of knowledge about the scene: this enables us to use the results at any phase of the algorithm and to choose the right balance between speed and precision without modifying the algorithm;
- (iii) real-time compliance, as the fitness function may be continuously updated in function of external parameters during execution.

Artificial evolution opens several interesting approaches to image processing, either by extending the scope of parametric methods (e.g., Hough) or through the introduction of its alternative programming philosophy—a way to easily and efficiently implement the very principles of artificial intelligence. Evolutionary methods in image processing use explicit references to models. This often allows to get free from the classical image processing toolbox (segmentation operators) to solve new problems with general-purpose, often more efficient tools. Additionally, the intrinsic asynchronous properties of artificial evolution offers new opportunities with real-time applications like the robot vision example described above.

This conclusion would not be complete if we did not briefly mention the fact that using evolutionary methods in model-based vision helps to extend the scope of machine vision itself. As image analysis can be defined as the task of rebuilding a model of reality from images taken by cameras, it may be interesting to quote work on the identification of mechanical models from image sequences. This very complex problem, connected to reverse engineering, has been solved in some particular applications thanks to multiobjective evolutionary computation. In one of them, mobile 3D objects could be reconstructed as passive mechanical structures whose internal parameters were identified using image sequence data [10, 11]. A similar evolutionary approach was used to identify the internal parameters of turbulent fluid flows [8] and mechanical structures with internal actuators (“muscular models”) [20] from image sequences. This time, the question is not merely the improvement of vision algorithms performance, but the very semantics of what has to be called “image analysis” or “machine vision.”

Acknowledgments

The author owes special thanks to Dr. Amine Boumaza, Dr. Pierre Collet, Baudoin Coppieters De Gibson, Anders Ekman, Dr. Jean-Loup Florens, Dr. Philippe Guermeur, Dr. Marie-Jeanne Lesot, Dr. Annie Luciani, Dr. Evelyne Lutton, Olivier Pauplin, Dr. Bogdan Stanciulescu, Dr. Michel Parent, and the COMPLEX team, who contributed to the methods, examples, and results given in this chapter. Special acknowledgments are due to Dr. Amine Boumaza who designed, implemented, and tested the simulator and the robot controllers described in Sections 14.3.3 and 14.3.4.

Bibliography

- [1] T. Bäck and H.-P. Schwefel, “Evolution strategies I: variants and their computational implementation,” in *Genetic Algorithms in Engineering and Computer Science*, pp. 111–126, John Wiley & Sons, Chichester, UK, 1995.

A print edition of this book can be purchased at

<http://www.hindawi.com/spc.8.html>

<http://www.amazon.com/dp/9774540018>

- [2] A. M. Boumaza and J. Louchet, "Dynamic flies: using real-time parisian evolution in robotics," in *Proceedings of Applications of Evolutionary Computing (EvoIASP '01)*, vol. 2037 of *Lecture Notes in Computer Science*, pp. 288–297, Como, Italy, April 2001.
- [3] P. Collet, E. Lutton, F. Raynal, and M. Schoenauer, "Individual GP: an alternative viewpoint for the resolution of complex problems," in *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO '99)*, vol. 2, pp. 974–981, Orlando, Fla, USA, July 1999.
- [4] R. Eberhart and J. Kennedy, "A new optimizer using particle swarm theory," in *Proceedings of the 6th International Symposium on Micro Machine and Human Science (MHS '95)*, pp. 39–43, IEEE Service Centre, Nagoya, Japan, October 1995.
- [5] R. C. Gonzalez and R. E. Woods, *Digital Image Processing*, John Wiley & Sons, New York, NY, USA, 1992.
- [6] R. M. Haralick, "Using perspective transformations in scene analysis," *Computer Graphics and Image Processing*, vol. 13, no. 3, pp. 191–221, 1980.
- [7] P. V. C. Hough, "Method and means of recognising complex patterns," US patent no. 3 069 654, December 1962.
- [8] J. Louchet and L. Jiang, "An identification tool to build physical models for virtual reality," in *Proceedings of the 3rd International Workshop on Image and Signal Processing (IWISP '96)*, pp. 669–672, Manchester, UK, November 1996.
- [9] Y. Koren and J. Borenstein, "Potential field methods and their inherent limitations for mobile robot navigation," in *Proceedings of IEEE International Conference on Robotics and Automation (ICRA '91)*, vol. 2, pp. 1398–1404, Sacramento, Calif, USA, April 1991.
- [10] J. Louchet, "An evolutionary algorithm for physical motion analysis," in *Proceedings of the Conference on British Machine Vision*, vol. 2, pp. 701–710, BMVA Press, York, UK, September 1994.
- [11] J. Louchet, X. Provot, and D. Crochemore, "Evolutionary identification of cloth animation models," in *Proceedings of the Eurographics Workshop on Computer Animation and Simulation*, pp. 44–54, Springer, Maastricht, The Netherlands, September 1995.
- [12] J. Louchet, "Using an individual evolution strategy for stereovision," *Genetic Programming and Evolvable Machines*, vol. 2, no. 2, pp. 101–109, 2001.
- [13] J. Louchet, M. Guyon, M.-J. Lesot, and A. Boumaza, "Dynamic flies: a new pattern recognition tool applied to stereo sequence processing," *Pattern Recognition Letters*, vol. 23, no. 1–3, pp. 335–345, 2002.
- [14] E. Lutton and P. Martinez, "A genetic algorithm for the detection of 2D geometric primitives in images," in *Proceedings of the 12th IAPR International Conference on Pattern Recognition (ICPR '94)*, vol. 1, pp. 526–528, Jerusalem, Israel, October 1994, INRIA Technical Report 2210.
- [15] J. O'Rourke, "Motion detection using Hough techniques," in *Proceedings of IEEE Computer Society Conference on Pattern Recognition and Image Processing*, pp. 82–87, Dallas, Tex, USA, August 1981.
- [16] I. Rechenberg, "Evolution strategy," in *Computational Intelligence: Imitating Life*, J. M. Zurada, R. J. Marks II, and C. Robinson, Eds., pp. 147–159, IEEE Press, Piscataway, NJ, USA, 1994.
- [17] G. Roth and M. D. Levine, "Geometric primitive extraction using a genetic algorithm," in *Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR '92)*, pp. 640–643, Champaign, Ill, USA, June 1992.
- [18] R. Salomon and P. Eggenberger, "Adaptation on the evolutionary time scale: a working hypothesis and basic experiments," in *Proceedings of the 3rd European Conference on Artificial Evolution (AE '97)*, vol. 1363 of *Lecture Notes in Computer Science*, pp. 251–262, Springer, Nîmes, France, October 1997.
- [19] P. K. Ser, C. S. T. Choy, and W. C. Siu, "Genetic algorithm for the extraction of nonanalytic objects from multiple dimensional parameter space," *Computer Vision and Image Understanding*, vol. 73, no. 1, pp. 1–13, 1999.
- [20] B. Stanciulescu, J.-L. Florens, A. Luciani, and J. Louchet, "Physical modeling framework for robotics applications," in *IEEE International Conference on Systems, Man and Cybernetics*, vol. 3, pp. 2213–2218, Washington, DC, USA, October 2003.

- [21] J. S. Zelek, "Complete real-time path planning during sensor-based discovery," in *Proceedings of the IEEE International Conference on Intelligent Robots and Systems*, vol. 3, pp. 1399–1404, Victoria, BC, Canada, October 1998.

Jean Louchet: APIS team, INRIA Futurs, Parc Orsay Università, 4 rue Jacques Monod, 91893 Orsay Cedex, France
Email: jean.louchet@m4x.org

15

Evolutionary feature synthesis for image databases

Anlei Dong, Bir Bhanu, and Yingqiang Lin

The high dimensionality of visual features is one of the major challenges for content-based image retrieval (CBIR) systems, and a variety of dimensionality reduction approaches have been proposed to find the discriminant features. In this chapter, we investigate the effectiveness of coevolutionary genetic programming (CGP) in synthesizing feature vectors for image databases from traditional features that are commonly used. The transformation for feature dimensionality reduction by CGP has two unique characteristics for image retrieval: (1) nonlinearity: CGP does not assume any class distribution in the original visual feature space; (2) explicitness: unlike kernel trick, CGP yields explicit transformation for dimensionality reduction so that the images can be searched in a low-dimensional feature space. The experimental results on multiple databases show that (a) CGP approach has distinct advantage over the linear transformation approach of multiple discriminant analysis (MDA) in the sense of the discrimination ability of the low-dimensional features, and (b) the classification performance using the features synthesized by our CGP approach is comparable to or even superior to that of support vector machine (SVM) approach using the original visual features.

15.1. Introduction

In recent years, the rapid advances in digital imaging technology and the low costs of cameras, scanners, and storage devices make large-size image databases possible. With explosive expanding of the Internet, efficient management of such image databases becomes necessary for many multimedia applications in the fields of business, education, and entertainment.

Traditional text-based image retrieval systems have some vital disadvantages. First, manually describing and tagging image contents in terms of a selected set of captions and keywords are too time-consuming, especially for large image databases. Second, textual description method is inadequate to describe image since an image is subjective to different users.

For the above reasons, content-based image retrieval (CBIR) is receiving widespread research interest [2, 5, 26, 29, 37]. Recent years have witnessed a variety of

content-based image retrieval systems: QBIC [9], Photobook [23], Virage [11], NeTra [20], MARS [24], VisualSEEk [30], PicHunter [6], and SIMPLIcity [38]. The images in these systems are represented by numeric values, such as texture, color, shape, and structure, which are called low-level visual features. The desired objects in people's mind are called human high-level concepts. The big gap between low-level features and high-level concepts necessitates learning method in the image retrieval system.

Content-based image retrieval systems are designed to automatically extract various visual features from images, such as color, texture, shape, structure, and use them to represent images in the computer. The collection of such visual features usually yields the high dimensionality of the feature space, which deteriorates retrieval performances due to the well-known "curse of dimensionality." Thus, a key task in CBIR is to find the most discriminant features from the original feature collection, so that both retrieval precision and search speed are improved.

Fisher discriminant analysis (also called multiple discriminant analysis (MDA) for multiple-class case) [8] is a widely used approach to find discriminating features due to its straight-forward idea of making linear data projection by maximizing the ratio of the *between scatter matrix* and the *within scatter matrix*. The kernel trick [28] can be combined with Fisher discriminant analysis (e.g., [3]), so that the nonlinear generalization of MDA is achieved with the improvement of classification.

Swets and Weng [33] propose a *self-organizing hierarchical optimal subspace learning and inference framework* (SHOSLIF) for image retrieval. The main idea is to recursively implement linear discriminant analysis on the data subsets obtained from the recursive subdivision of the data, so that the limitation of the global linear transformation is overcome. Such hierarchical linear analysis is a nonlinear approach in nature.

Hastie and Tibshirani [12] present the approach of *discriminant adaptive nearest neighbor* (DANN) for classification. Based on the analysis of local dimension information, they also propose a global dimensionality method by pooling local discriminant information.

Tieu and Viola [35] propose the AdaBoost algorithm to learn a strong classifier for image retrieval. The strong classifier consists of some weak classifiers corresponding to some original visual features, and it gives out retrieval results fast and efficiently.

Wu et al. [39] reduce the feature dimensionality for image databases using the approach of *weighted multidimensional scaling* (WMDS), whose main characteristic is preserving the local topology of the high-dimensional space. Su et al. [32] exploit principal component analysis (PCA) for dimensionality reduction by using relevance feedback.

He et al. [13] regard the data space as a manifold, and use locality preserving projection (LPP) to preserve the local structure of the image space [13]. It has been proved that LPP has more discriminating power than PCA.

In this chapter, we investigate the effectiveness of coevolutionary genetic programming (CGP) [16] in generating composite operator vectors for image

databases, so that feature dimensionality is reduced to improve retrieval performances. Genetic programming (GP) is an evolutionary computational paradigm, that is, an extension of genetic algorithm and works with a population of individuals. An individual in a population can be any complicated data structure such as linked lists, trees, and graphs. CGP is an extension of GP in which several populations are maintained and employed to evolve solutions cooperatively. A population maintained by CGP is called a subpopulation and it is responsible for evolving a part of a solution. A complete solution is obtained by combining the partial solutions from all the subpopulations. For the task of object recognition in [19], individuals in subpopulations are composite operators, which are the elements of a composite operator vector. A composite operator is represented by a binary tree whose internal nodes are the prespecified domain-independent primitive operators and leaf nodes are original features. It is a way of combining original features. The CGP approach in this chapter follows the algorithm proposed in [19].

The advantage of using a tree structure is that it is powerful enough in expressing the ways of combining original features. Unlike a graph, it has no loops and this guarantees that the execution of individuals represented by trees will terminate and not be trapped in an infinite loop. The original features are visual features (e.g., color, texture, structure) extracted from the images. With each element evolved by a subpopulation of CGP, a composite operator vector is cooperatively evolved by all the subpopulations. By applying composite operators (corresponding to each subpopulation) to the original features extracted from images, composite feature vectors are obtained. These composite feature vectors are fed into a classifier for recognition.

The transformation for feature dimensionality reduction by CGP has two unique characteristics that are suitable for image retrieval: (1) *nonlinearity*: CGP does not assume any class distribution in the original visual feature space; (2) *explicitness*: unlike kernel trick, CGP yields explicit transformation for dimensionality reduction. Thus, during online image search, the system only needs to compute the features in this reduced-dimensional feature space, instead of the original higher-dimensional feature space, and this significantly reduce search time.

The contributions of this chapter are (1) the coevolutionary genetic programming (CGP) approach in generating composite operator vectors for image databases, so that the feature dimensionality is reduced and the retrieval performance is improved (Section 15.3); (2) the comparisons of the experimental results by CGP, MDA, and SVM on multiple image databases, so that the advantage of the CGP approach is demonstrated (Section 15.4).

15.2. Related research on GP and CGP

In general, feature selection and feature synthesis are two kinds of feature transformations. In feature selection, original features are not changed and some original features are selected to form a subset of features to be used by classifiers. Genetic algorithm is widely used in feature selection [4]. In feature synthesis, a transformation, linear or nonlinear, is applied to the original features to generate new

features. Weighted summation is a kind of linear transformation on the original features, and the weights of features can be determined by a genetic algorithm. In multilayer neural networks, each node of a neural network takes the weighted sum of the outputs of its child nodes as input [34]. The weights are determined by backpropagation algorithm during training. The output of a node is determined by the input and the activation function of the node. It can be viewed as a non-linear transformation on the original features. The CGP-based feature synthesis is another kind of nonlinear transformation on the original features, which are the primitive features in this paper.

The primitive operators in this paper are not logical operators, but operators on real numbers and the composite operators are binary trees of primitive operators on real numbers not binary trees of logical operators. In [31], GP is used to evolve logical expressions and the final outcome of the logical expression determines the type of the object under consideration (e.g., 1 means target and 0 means clutter). In this paper, CGP is used to evolve composite feature vectors to be used by a Bayesian classifier [34] and each subpopulation is responsible for evolving a specific composite feature in the composite feature vector. The classifier evolved by GP in [31] is a logical expression represented by the binary tree with the best classification rate in the population, but the classifier evolved by CGP in this paper is a Bayesian classifier determined by the composite feature vectors obtained from the training images. Unlike the work of Krawiec and Bhanu [17], composite operators in this paper are binary trees of primitive operators and primitive features, whereas the recognition procedures in [17] are linked lists of simple image processing operations.

15.3. Technical approach

15.3.1. Labeling scenario

Due to the high feature dimensionality, content-based image retrieval systems usually cannot achieve satisfactory retrieval performance purely with image visual features which are automatically extracted from images. The CBIR systems have to interact with humans to obtain some labeling information, which is then used for learning or adaptation. There are three possible ways for a retrieval system to obtain labeling information: (1) labeled images for training are provided in advance (supervised learning); (2) allow a user to execute relevance feedback for his/her retrieval [25] (short-term semisupervised learning) [24, 22, 27]; (3) accumulate multiple users' retrieval experiences to learn concepts in long term (long-term semisupervised learning) [7, 10, 14, 15, 18, 40–42]. Since the second method, relevance feedback for a single user, is only a short-term adaptation to the specific requirement of this user, it cannot help to find the global discriminant features.

In this chapter, with the purpose of demonstrating the effectiveness of coevolutionary genetic programming approach for image databases, we simply adopt the first scenario, that is, training the system by directly providing labeled images

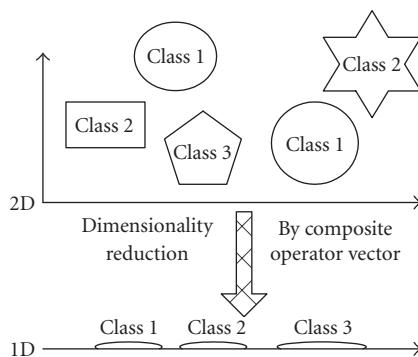


FIGURE 15.1. An illustrative example for feature reduction (from 2D to 1D) using composite operation. The original 2D feature space: three classes have different distributions, and Class 1 and Class 2 consist of multiple clusters. The transformed 1D feature space: each of the three classes corresponds to a single Gaussian distribution.

in advance. In future, we can also extend the proposed method to the scenario of processing the multiple users' retrieval experiences.

15.3.2. Image distribution

By intuition, the images belonging to the same class should be close in some sense in the feature space of the image database, that is, these images form a cluster with a specific shape, which reflects the relevances of the various visual features for the corresponding class. However, we cannot ignore the possibility that the images in the same class (as perceived by the user) may form multiple clusters, that is, the images with different visual features may belong to the same class.

Based on the above observations, some researchers have proposed the Gaussian mixture model for the image distribution in the feature space of the image databases [36], with each class corresponding to a single mixture component or a variety of mixture components. Sometimes, mixture model assumption for the image distribution of a database may be too strict due to the existence of *outliers* (the images are very far away from the components they belong to) and *clutter* (the images that do not belong to any component). In this chapter, the proposed CGP approach does not make any model assumption for the image distribution, so that the negative effects such as overfitting by the model-based approaches can be avoided or alleviated.

Figure 15.1 provides an example which illustrates the characteristics of our CGP approach for reducing the feature dimensionality of image databases. The original 2D data belong to three classes, two of which (Class 1 and Class 2) form multiple clusters. Different clusters have different distributions (represented by different shapes). The transformation by the CGP approach yields 1D feature space, in which each of the three classes corresponds to a single Gaussian distribution.

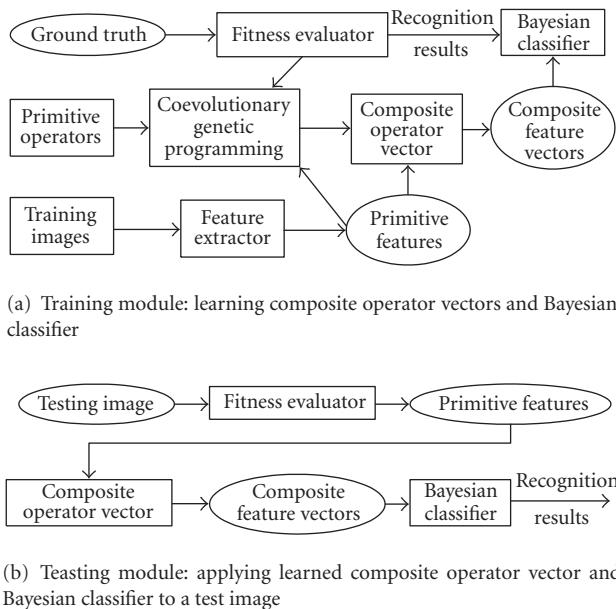


FIGURE 15.2. System diagram for object recognition using coevolutionary genetic programming.

15.3.3. Coevolutionary genetic programming

In CGP approach, each synthesized feature is derived by implementing a series of operators on the original visual features. Such operators are called composite operators, which are represented by binary trees with primitive operators as internal nodes and original features as leaf nodes. The goal of these composite operators is to map the original visual feature space to the low-dimensional synthesized feature space, in which the images belonging to the same class form a Gaussian component no matter how these images are distributed in the original visual space.

The search space of all possible composite operators is so huge that it is extremely difficult to find good composite operators from this vast space unless one has a smart search strategy. How to design such a smart search strategy is the task for CGP algorithm. We follow the CGP algorithm proposed in [19], which attempts to improve the performance of object recognition. Figure 15.2 shows the training and testing modules of the system. During training, CGP runs on training images and evolves composite operators to obtain composite features. Since a Bayesian classifier is derived from the feature vectors obtained from training images, both the composite operator vector and the classifier are learned by CGP.

The set of primitive operators. A primitive operator takes one or two real numbers, performs a simple operation on them and outputs the result. Table 15.1 shows the 12 primitive operators being used, where a and b are real numbers and input to an operator and c is a constant real number stored in an operator.

TABLE 15.1. Twelve primitive operators.

Primitive operator	Description
ADD (a, b)	Add a and b .
ADDC (a, c)	Add constant value c to a .
SUB (a, b)	Subtract b from a .
SUBC (a, c)	Subtract constant value c from a .
MUL (a, b)	Multiply a and b .
MULC (a, c)	Multiply a with constant value c .
DIV (a, b)	Divide a by b .
DIVC (a, c)	Divide a by constant value c .
MAX2 (a, b)	Get the larger of a and b .
MIN2 (a, b)	Get the smaller of a and b .
SQRT (a)	Return \sqrt{a} if $a \geq 0$; otherwise, return $-\sqrt{-a}$.
LOG (a)	Return $\log(a)$ if $a \geq 0$; otherwise, return $-\log(-a)$.

Fitness measure. The fitness of a composite operator vector is computed in the following way: apply each composite operator of the composite operator vector on the original features of training images to obtain composite feature vectors of training images and feed them to a Bayesian classifier. Note that not all the original features are necessarily used in feature synthesis. Only the original features that appear in the leaf nodes of the composite operator are used to generate composite features. The recognition rate of the classifier is the fitness of the composite operator vector. To evaluate a composite operator evolved in a subpopulation, the composite operator is combined with the current best composite operators in other subpopulations to form a complete composite operator vector where composite operator from the i th subpopulation occupies the i th position in the vector and the fitness of the vector is defined as the fitness of the composite operator under evaluation. The fitness values of other composite operators in the vector are not affected. When subpopulations are initially generated, the composite operators in each subpopulation are evaluated individually without being combined with composite operators from other subpopulations. In each generation, the composite operators in the first subpopulation are evaluated first, then the composite operators in the second subpopulation, and so on.

Parameters and termination. The key parameters are the number of subpopulations N , the population size M , the number of generations G , the crossover and mutation rates, and the fitness threshold. GP stops whenever it finishes the specified number of generations or the performance of the Bayesian classifier is above the fitness threshold. After termination, CGP selects the best composite operator

of each subpopulation to form the learned composite operator vector to be used in testing.

Selection, crossover, and mutation. The CGP searches through the space of composite operator vectors to generate new composite operator vectors. The search is performed by selection, crossover, and mutation operations. The initial subpopulations are randomly generated. Although subpopulations are cooperatively evolved (the fitness of a composite operator in a subpopulation is not solely determined by itself, but affected by the composite operators from other subpopulations), selection is performed only on composite operators within a subpopulation and crossover is not allowed between two composite operators from different subpopulations.

Selection. The selection operation involves selecting composite operators from the current subpopulation. In this chapter, tournament selection is used and the tournament size is empirically selected as 5. The higher the fitness value is, the more likely the composite operator is selected to survive.

Crossover. Two composite operators, called parents, are selected on the basis of their fitness values. The higher the fitness value is, the more likely the composite operator is selected for crossover. One internal node in each of these two parents is randomly selected, and the two subtrees rooted at these two nodes are exchanged between the parents to generate two new composite operators called offspring. It is easy to see that the size of one offspring (i.e., the number of nodes in the binary tree representing the offspring) may be greater than both parents if crossover is implemented in such a simple way. To prevent code bloat, we specify a maximum size of a composite operator (called max-operator-size). If the size of one offspring exceeds the max-operator-size, the crossover is performed again. If the size of an offspring still exceeds the max-operator-size after the crossover is performed 10 times, GP selects two subtrees of the same size (i.e., the same number nodes) from two parents and swaps the subtrees between the parents. These two subtrees can always be found, since a leaf node can be viewed as a subtree of size 1.

Mutation. To avoid premature convergence, mutation is introduced to randomly change the structure of some composite operators to maintain the diversity of subpopulations. Candidates for mutation are randomly selected and the mutated composite operators replace the old ones in the subpopulations. There are three mutations invoked with equal probability.

- (1) Randomly select a node of the composite operator and replace the subtree rooted at this node by another randomly generated binary tree.
- (2) Randomly select a node of the composite operator and replace the primitive operator stored in the node with another primitive operator randomly selected from the primitive operators of the same number of input as the replaced one.
- (3) Randomly select two subtrees of the composite operator and swap them. Of course, neither of the two subtrees can be a subtree of the other.

15.3.4. Generational coevolutionary genetic programming

Generational coevolutionary genetic programming is used to evolve composite operators. The generational coevolutionary genetic programming algorithm [19] is shown in Algorithm 15.1. The GP operations are applied in the order of crossover, mutation, and selection. The composite operators in the initial subpopulations are randomly generated. A composite operator is generated in two steps. In the first step, the number of internal nodes of the tree representing the composite operator is randomly determined as long as this number is smaller than half of max-operator-size. Suppose the tree has n internal nodes. The tree is generated from top to bottom by a tree generation algorithm. The root node is generated first and the primitive operator stored in the root node is randomly selected. The selected primitive operator determines the number of children the root node has. If it has only one child, the algorithm is recursively invoked to generate a tree of $n - 1$ internal nodes; if it has two children, the algorithm is recursively invoked to generate two trees of $(n - 1)/2$ and $(n - 1)/2$ internal nodes, respectively. In the second step, after all the internal nodes are generated, the leaf nodes containing original features are attached to those internal nodes that are temporarily the leaf nodes before the real leaf nodes are attached. The number of leaf nodes attached to an internal node is determined by the primitive operator stored in the internal node. In addition, an elitism replacement method is adopted to keep the best composite operator from generation to generation.

To evaluate C_j for Step (6) in Algorithm 15.1, select the current best composite operator in each of the other subpopulations, combine C_j with those $N - 1$ best composite operators to form a composite operator vector where composite operator from the k th subpopulation occupies the k th position in the vector ($k = 1, \dots, N$). Run the composite operator vector on the original features of the training images to get composite feature vectors and use them to build a Bayesian classifier. Feed the composite feature vectors into the Bayesian classifier and let the recognition rate be the fitness of the composite operator vector and the fitness of C_j .

15.3.5. Indexing structure

For each class C_i , a Bayesian classifier is generated based on GP-learned composite features. In the low-dimensional feature space, each class is corresponding to a single Gaussian component, which is represented by the mean feature vector and the covariance matrix of feature vectors of this class. Note that the synthesized features in the low-dimensional space are independent; thus, the covariance matrices for all the classes are diagonal.

We construct the indexing structure based on the Gaussian components obtained by CGP approach. When a query image comes, the system computes the probabilities that it belongs to those components using the components' parameters (means and covariances), and executes the search within the component with the highest probability.

```

(1) Randomly generate  $N$  subpopulations of size  $M$  and evaluate each composite
operator in each subpopulation individually.

for gen = 1 to generation_num do
  for  $i$  = 1 to  $N$  do
    (1) Implement standard unsupervised EM algorithm on training data  $\mathcal{X}$ .
    (2) Keep the best composite operator in subpopulation  $P_i$ .
    (3) Perform crossover on the composite operators in  $P_i$  until the
        crossover rate is satisfied and keep all the offspring from crossover.
    (4) Perform mutation on the composite operators in  $P_i$  and the offspring
        from crossover with the probability of mutation rate.
    (5) Perform selection on  $P_i$  to select some composite operators and com-
        bine them with the composite operators from crossover to get a new
        subpopulation  $P'_i$  of the same size as  $P_i$ .
    (6) Evaluate each composite operator  $C_j$  in  $P'_i$ .
    (7) Perform elitism replacement.
    (8) Form the current best composite operator vector consisting of the
        best composite operators from corresponding subpopulations and
        evaluate it. If its fitness is above the fitness threshold, go to 2.
  end for
end for

(2) Select the best composite operator from each subpopulation to form the learned
composite operator vector and output it.

```

ALGORITHM 15.1. Generational coevolutionary genetic programming.

The indexing structure implies two advantages for image retrieval: (1) the search is carried on only in the subset of the whole image database; (2) the search is carried on in the low-dimensional feature space. Both of these make the search faster; furthermore, our CGP approach guarantees that the retrieval precision is high since the Bayesian classification is good in the low-dimensional feature space. Thus, the retrieval performance is improved by our CGP approach.

15.4. Experiments

To evaluate the effectiveness of the CGP approach for reducing the feature dimensionality of image databases, we implement the CGP algorithm on three different image databases with the purpose to evaluate its effectiveness for different kinds of class distribution. All the images in these three databases are selected from Corel stock photo library [1], which contains 200 CDs and each CD has 100 images. We also compare the results of the CGP approach with other approaches including MDA and SVM.

15.4.1. Image databases

(1) *DB1200*: We construct an image database with 1200 images, which are selected from Corel stock photo library and divided into 12 classes. These classes are

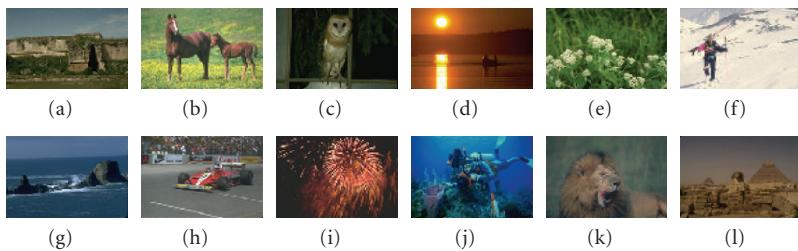


FIGURE 15.3. Sample images of the 12 classes in the database obtained from Corel stock photo library.

corresponding to the CDs (series number) in the library including *Mayan & Aztec Ruins* (33), *Horses* (113), *Owls* (75), *Sunrises & Sunsets* (1), *North American Wild flowers* (127), *Ski Scenes* (61, 62), *Coasts* (5), *Auto Racing* (21), *Firework Photography* (73), *Divers & Diving* (156), *Land of the Pyramids* (161), and *Lions* (105). We remove some images from these CDs since they do not have good visual features to represent the corresponding concepts, and we add some images from other CDs to some of the 12 classes. Figure 15.3 shows sample images for all of the 12 concepts. We use texture and color features to represent images. The texture features are derived from 16 Gabor filters [22]. We extract means and standard deviations from the three channels in HSV color space. Thus, each image is represented by 22 features.

(2) *DB1500*: We add other 300 images (from other three CDs in the Library) into DB1200 to obtain DB1500. The three new CDs (series number) are *hawks and falcons* (70 000), *tigers* (108 000), and *tulips* (258 000). Each of these three CDs is merged to one existing CDs in DB1200 to form a class, so that DB1500 still has 12 classes. In these 12 classes, there are three classes each of which consists of two clusters in visual feature space: the CD of *Hawks and Falcons* and the CD of *owls* form the class of *bird*, the class of *tulips* and the class of *North American wild flowers* form the class of *flowers*, and the class of *tigers* and the class of *lions* form the concept of *wild beasts*. Figure 15.4 shows the sample images of these three classes containing multiple CDs. Obviously, DB1500 is challenging for the linear transformation approach such as MDA, as we will show later in the experiments.

(3) *DB6600*: The 6 600 images are obtained from 66 CDs, which are assigned to 50 classes,¹ that is, each class may consist of a single CD or multiple CDs, for

¹The 50 classes in DB6600 are corresponding to the CDs (series number) in Corel stock photo library including (1) *action sailing* (172 000) + *sailboats* (7000), (2) *African antelope* (77 000) + *African specialty animals* (130 000), (3) *air shows* (10 000) + *aviation photography* (34 000) + *WWII planes* (3000), (4) *annuals for American gardens* (132 000) + *flowering potted plants* (124 000) + *perennials in bloom* (133 000) + *roses* (84 000) + *flowers* (13 000) (5) *apes* (49 000), (6) *Arabian horses* (113 000) + *horses* (197 000), (7) *auto racing* (21 000) + *exotic cars* (29 000), (8) *autumn* (150 000), (9) *bald eagles* (135 000) + *hawks & falcons* (70 000), (10) *bears* (100 000), (11) *beneath the Caribbean* (141 000), (12) *bridges* (22 000), (13) *butterflies* (52 000), (14) *cactus flowers* (51 000), (15) *candy backgrounds* (96 000), (16) *Caribbean* (68 000), (17) *caves* (194 000), (18) *churches* (24 000), (19) *cities of Italy* (111 000), (20) *coasts* (5000), (21) *coins & currency* (125 000), (22) *Death Valley* (39 000), (23) *divers & diving* (156 000), (24) *dogs* (247 000), (25) *doors of San Francisco* (59 000), (26) *elephants*

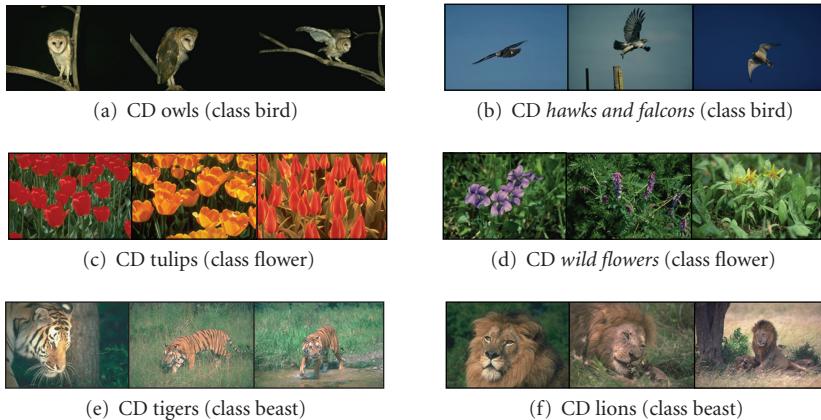


FIGURE 15.4. DB1500: sample images from the three classes containing multiple CDs.

instance, the CDs *Arabian horses*. And *horses* are assigned to the same class since they are both for the same concept of *horse*. On the other hand, even the images within the same CD may form multiple clusters in the sense of visual features. For example, the CD of *cities of Italy* consists of images for different objects such as building, road, people. For most of the classes, there are many outlier images, each of whose visual features is far away from the cluster(s) its corresponding class contains.

Original visual features: Images are represented by texture features, color features, and structure features. The texture features are derived from 16 Gabor filters [21]. We also extract means and standard deviations from the three channels in HSV color space. For structure features, we use the water-filling approach [43] to extract 18 features from each image. Thus, each image is represented by 40 visual features.

15.4.2. Experiments results

Experimental parameters. For each of the three image databases, we randomly select half of the images as training data and another half as testing data. We implement CGP algorithm on the training images, and the parameters values are: (a) subpopulation size: 50; (b) crossover rate: 0.6; (c) number of generation: 50;

(107 000), (27) *English country gardens* (131 000), (28) *fields* (28 000), (29) *firework photography* (73 000) + *fireworks* (40 000), (30) *glaciers & mountains* (114 000) + *mountains of America* (2000), (31) *ice & icebergs* (184 000), (32) *indigenous people* (189 000), (33) *insects* (35 000), (34) *landscapes* (176 000), (35) *lions* (105 000), (36) *Mayan & Aztec ruins* (33 000), (37) *mushrooms* (158 000), (38) *North American wildflowers* (127 000), (39) *ocean life* (164 000), (40) *oil paintings* (211 000), (41) *owls* (75 000), (42) *polar bears* (183 000), (43) *residential interiors* (31 000), (44) *rhinos & hippos* (112 000), (45) *Rome* (149 000), (46) *ski scenes* (61 000) + *skiing in Switzerland* (60 000), (47) *reptiles and amphibians* (87 000) + *snakes, lizards & salamanders* (175 000), (48) *sundsets and sunrises* (1000), (49) *water falls* (27 000). (50) *religious stained glass* (99 000).

TABLE 15.2. Classification errors: the comparison of the three different approaches on different databases.

Database	CGP (10D)	MDA (10D)	SVM (40D)
DB1200	14.8%	25.5%	12.6%
DB1500	16.5%	31.7%	29.6%
DB6600	51.8%	73.3%	52.6%

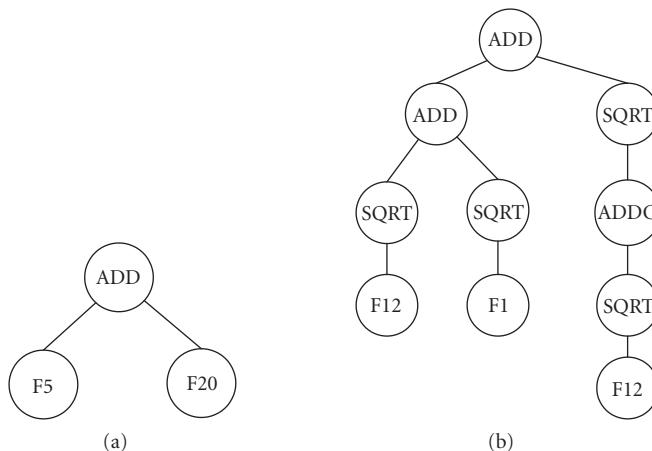


FIGURE 15.5. Sample composite operator vectors in DB6600 (the leaf nodes are original visual features): (a) a simple composite operator vector, (b) a complex composite operator vector.

(d) mutation rate: 0.05; (e) fitness threshold: 1.0; (f) tournament size: 5. For both CGP approach and MDA approach, we reduce the feature dimensionality from 40 to 10, so that it is fair to compare these two approaches.

Figure 15.5 gives two sample composite operator vectors in DB6600. The complex case in (b) illustrates that the unconventional combination of the operators exist in CGP, which may help to achieve good classification performance.

Classification performance. Table 15.2 shows the classification performances of the approaches of CGP, MDA, and SVM. The first two approaches attempt to reduce the feature dimensionality in *explicit* ways, so that data in the same class form a single Gaussian component in the lower-dimensional feature space. Thus, we present their Bayesian classification errors in the lower-dimensional (10D) feature space. Since SVM exploits kernel trick instead of providing explicit transformation, we present its classification error in the original visual feature space (40D).

From Table 15.2, we observe that the classification performance of CGP is better than that of MDA on all of the three databases. It is easy to understand the significant advantage of CGP over MDA on DB1500 and DB6600, both of which contain some classes which consist of multiple clusters in the original visual feature space. Therefore, the linear approach of MDA cannot deal with them well.

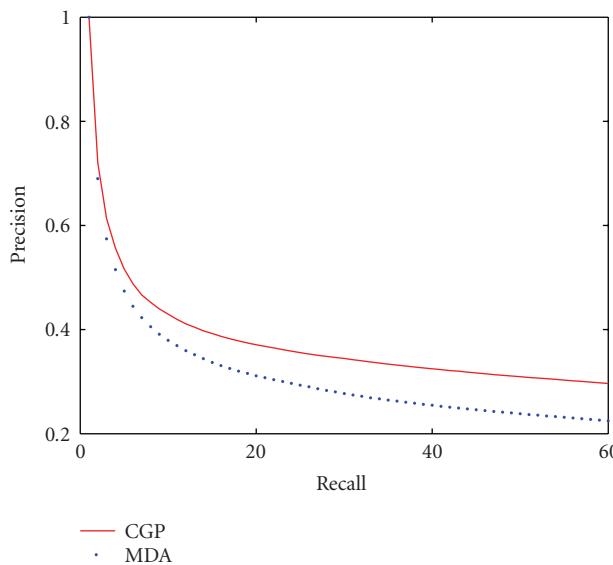


FIGURE 15.6. DB6600: precision-recall curves: CGP versus MDA.

Although each class in DB1200 only corresponds to a single cluster in the original visual feature space, CGP still outperforms MDA (14.8% vs. 12.6%) due to the reason that the distribution of these clusters may not be Gaussian while MDA assumes the cluster distribution as Gaussian and CGP does not make such an assumption.

We now compare CGP and SVM, which have similar classification errors on DB1200 and DB6600. However, the error (16.5%) by CGP is significantly lower to that (29.6%) by SVM on DB1500, which implies that CGP is more suitable to deal with the class which consists of multiple clusters in the original feature space. With many outliers in DB6600, such an advantage of CGP is not so obvious as in the case of DB1500 (CGP: 51.8% versus MDA: 52.6%), and neither CGP nor MDA can yield classification performance which is as good as those on DB1500.

Retrieval performance. After reducing the feature space, we construct the indexing structure based on the Gaussian components obtained by CGP approach or MDA approach.

We use each image in the database as query, and obtain average retrieval precisions. Figure 15.6 shows the retrieval-precision curves on DB6600 by CGP and MDA. We observe that CGP yields better retrieval results than MDA.

From the experiments reported above, we conclude that (a) feature synthesizing by CGP has obvious advantage over MDA in the sense of both classification and retrieval. (b) The classification performance in the synthesized feature space by CGP is comparable to or even superior to that in the original feature space by SVM. (c) Furthermore, the retrieval performance in the low-dimensional feature space is improved since the retrieval precision is higher and the search is faster.



FIGURE 15.7. DB6600: retrieval examples by CGP and MDA with the same query image (the first image in each group). The user seeks *planes*. (a) MDA: the nonplane images are row 1: image 3, 4; row 2 : 1; row 3 : 5; row 4: image 1, 2, 3, 5. (b) CGP: The only nonplane image is the last one in row 4.

Figure 15.7 presents a retrieval example, which illustrates that CGP approach improves the retrieval performance in the sense of higher precision and faster search.

15.5. Conclusions

This chapter presents a coevolutionary genetic programming (CGP) for feature dimensionality reduction. The two characteristics of the transformation by CGP, that is, nonlinearity and explicitness, make it suitable for image retrieval: (a) the nonlinearity yields good classification performance without any class distribution model assumption, and (b) explicitness achieves the image search in the low-dimensional feature space. Experimental results on multiple image databases with different types of distributions have demonstrated the effectiveness of improving image retrieval performance by the CGP approach.

Bibliography

- [1] <http://www.corel.com/>.
- [2] S. Antani, R. Kasturi, and R. Jain, "A survey on the use of pattern recognition methods for abstraction, indexing and retrieval of images and video," *Pattern Recognition*, vol. 35, no. 4, pp. 945–965, 2002.
- [3] G. Baudat and F. Anouar, "Generalized discriminant analysis using a kernel approach," *Neural Computation*, vol. 12, no. 10, pp. 2385–2404, 2000.
- [4] B. Bhanu and Y. Lin, "Genetic algorithm based feature selection for target detection in SAR images," *Image and Vision Computing*, vol. 21, no. 7, pp. 591–608, 2003.
- [5] V. Castelli and L. Bergman, *Image Databases: Search and Retrieval of Digital Imagery*, John Wiley & Sons, New York, NY, USA, 2002.
- [6] I. J. Cox, M. L. Miller, T. P. Minka, T. V. Papathomas, and P. N. Yianilos, "The Bayesian image retrieval system, *PicHunter*: theory, implementation, and psychophysical experiments," *IEEE Transactions on Image Processing*, vol. 9, no. 1, pp. 20–37, 2000.
- [7] A. Dong and B. Bhanu, "Active concept learning for image retrieval in dynamic databases," in *Proceedings of the 9th IEEE International Conference on Computer Vision (ICCV '03)*, vol. 1, pp. 90–95, Nice, France, October 2003.
- [8] R. Duda, P. Hart, and D. Stork, *Pattern Classification*, John Wiley & Sons, New York, NY, USA, 2nd edition, 2001.
- [9] M. Flickner, H. Sawhney, W. Niblack, et al., "Query by image and video content: the QBIC system," *Computer*, vol. 28, no. 9, pp. 23–32, 1995.
- [10] J. Fournier and M. Cord, "Long-term similarity learning in content-based image retrieval," in *Proceedings of the IEEE International Conference on Image Processing (ICIP '02)*, vol. 1, pp. 441–444, Rochester, NY, USA, September 2002.
- [11] A. Hampapur, A. Gupta, B. Horowitz, et al., "Virage image search engine: an open framework for image management," in *Storage and Retrieval for Image and Video Databases V*, vol. 3022 of *Proceedings of SPIE*, pp. 188–198, San Jose, Calif, USA, February 1997.
- [12] T. Hastie and R. Tibshirani, "Discriminant adaptive nearest neighbor classification," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 18, no. 6, pp. 607–616, 1996.
- [13] X. He, S. Yan, Y. Hu, and H.-J. Zhang, "Learning a locality preserving subspace for visual recognition," in *Proceedings of the 9th IEEE International Conference on Computer Vision (ICCV '03)*, vol. 1, pp. 385–392, Nice, France, October 2003.
- [14] D. R. Heisterkamp, "Building a latent semantic index of an image database from patterns of relevance feedback," in *Proceedings of the 16th International Conference on Pattern Recognition (ICPR '02)*, vol. 4, pp. 134–137, Quebec, Canada, August 2002.
- [15] T. Hertz, N. Shental, A. Bar-Hillel, and D. Weinshall, "Enhancing image and video retrieval: learning via equivalence constraints," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR '03)*, vol. 2, pp. 668–674, Madison, Wis, USA, June 2003.

- [16] J. Koza, *Genetic Programming II: Automatic Discovery of Reusable Programs*, MIT Press, Cambridge, Mass, USA, 1994.
- [17] K. Krawiec and B. Bhanu, "Visual learning by evolutionary feature synthesis," in *Proceedings of the 20th International Conference on Machine Learning (ICML '03)*, vol. 1, pp. 376–383, Washington, DC, USA, August 2003.
- [18] M. Li, Z. Chen, and H.-J. Zhang, "Statistical correlation analysis in image retrieval," *Pattern Recognition*, vol. 35, no. 12, pp. 2687–2693, 2002.
- [19] Y. Lin and B. Bhanu, "Evolutionary feature synthesis for object recognition," *IEEE Transactions on Systems, Man and Cybernetics, Part C*, vol. 35, no. 2, pp. 156–171, 2005.
- [20] W. Y. Ma and B. S. Manjunath, "NeTra: a toolbox for navigating large image databases," in *Proceedings of the IEEE International Conference on Image Processing (ICIP '97)*, vol. 1, pp. 568–571, Santa Barbara, Calif, USA, October 1997.
- [21] B. S. Manjunath and W. Y. Ma, "Texture features for browsing and retrieval of image data," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 18, no. 8, pp. 837–842, 1996.
- [22] J. Peng, B. Bhanu, and S. Qing, "Probabilistic feature relevance learning for content-based image retrieval," *Computer Vision and Image Understanding*, vol. 75, no. 1-2, pp. 150–164, 1999.
- [23] A. P. Pentland, R. W. Picard, and S. Sclaroff, "Photobook: tools for content-based manipulation of image databases," in *23rd AIPR Workshop: Image and Information Systems: Applications and Opportunities*, vol. 2368 of *Proceedings of SPIE*, pp. 37–50, Washington, DC, USA, October 1995.
- [24] Y. Rui, T. S. Huang, and S. Mehrotra, "Content-based image retrieval with relevance feedback in MARS," in *Proceedings of the IEEE International Conference on Image Processing (ICIP '97)*, vol. 2, pp. 815–818, Santa Barbara, Calif, USA, October 1997, <http://www-db.ics.uci.edu/pages/research/mars.shtml/>.
- [25] Y. Rui, T. S. Huang, M. Ortega, and S. Mehrotra, "Relevance feedback: a power tool for interactive content-based image retrieval," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 8, no. 5, pp. 644–655, 1998.
- [26] Y. Rui, T. S. Huang, and S.-F. Chang, "Image retrieval: current techniques, promising directions, and open issues," *Journal of Visual Communication and Image Representation*, vol. 10, no. 1, pp. 39–62, 1999.
- [27] G. Salton, *Automatic Text Processing*, Addison-Wesley, Boston, Mass, USA, 1989.
- [28] B. Schölkopf and A. J. Smola, *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*, MIT Press, Cambridge, Mass, USA, 2001.
- [29] A. W. M. Smeulders, M. Worring, S. Santini, A. Gupta, and R. Jain, "Content-based image retrieval at the end of the early years," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 22, no. 12, pp. 1349–1380, 2000.
- [30] J. R. Smith and S.-F. Chang, "VisualSEEK: a fully automated content-based image query system," in *Proceedings of the 4th ACM International Multimedia Conference and Exhibition*, pp. 87–98, Boston, Mass, USA, November 1996.
- [31] S. A. Stanshope and J. M. Daida, "Genetic programming for automatic target classification and recognition in synthetic aperture radar imagery," in *Evolutionary Programming VII*, pp. 735–744, Springer, New York, NY, USA, 1998.
- [32] Z. Su, S. Li, and H. Zhang, "Extraction of feature subspaces for content-based retrieval using relevance feedback," in *Proceedings of the 9th ACM International Multimedia Conference and Exhibition*, pp. 98–106, Ottawa, Ontario, Canada, September-October 2001.
- [33] D. L. Swets and J. Weng, "Hierarchical discriminant analysis for image retrieval," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 21, no. 5, pp. 386–401, 1999.
- [34] S. Theodoridis and K. Koutroumbas, *Pattern Recognition*, Academic Press, New York, NY, USA, 1999.
- [35] K. Tieu and P. Viola, "Boosting image retrieval," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR '00)*, vol. 1, pp. 228–235, Hilton Head Island, SC, USA, June 2000.
- [36] N. Vasconcelos, *Bayesian models for visual information retrieval*, Ph.D. thesis, Massachusetts Institute of Technology, Cambridge, Mass, USA, 2000.

- [37] N. Vasconcelos and M. Kunt, "Content-based retrieval from image databases: current solutions and future directions," in *Proceedings of IEEE International Conference on Image Processing (ICIP '01)*, vol. 3, pp. 6–9, Thessaloniki, Greece, October 2001.
- [38] J. Z. Wang, J. Li, and G. Wiederhold, "SIMPLIcity: semantics-sensitive integrated matching for picture libraries," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 23, no. 9, pp. 947–963, 2001.
- [39] P. Wu, B. S. Manjunath, and H. D. Shin, "Dimensionality reduction for image retrieval," in *Proceedings of IEEE International Conference on Image Processing (ICIP '00)*, vol. 3, pp. 726–729, Vancouver, BC, Canada, September 2000.
- [40] P.-Y. Yin, B. Bhanu, K.-C. Chang, and A. Dong, "Improving retrieval performance by long-term relevance information," in *Proceedings of the 16th International Conference on Pattern Recognition (ICPR '02)*, vol. 3, pp. 533–536, Quebec, Canada, August 2002.
- [41] P.-Y. Yin, B. Bhanu, K.-C. Chang, and A. Dong, "Reinforcement learning for combining relevance feedback techniques," in *Proceedings of the 9th IEEE International Conference on Computer Vision (ICCV '03)*, vol. 1, pp. 510–515, Nice, France, October 2003.
- [42] X. Zhou, Q. Zhang, L. Liu, L. Zhang, and B. Shi, "An image retrieval method based on analysis of feedback sequence log," *Pattern Recognition Letters*, vol. 24, no. 14, pp. 2499–2508, 2003.
- [43] X. S. Zhou, Y. Rui, and T. S. Huang, *Exploration of Visual Data*, Kluwer Academic Publishers, Boston, Mass, USA, 2003.

Anlei Dong: Hermes-Microvision, Inc., Milpitas, San Jose, CA 95131, USA

Email: anlei.dong@hermes-microvision.com

Bir Bhanu: Center for Research in Intelligent Systems, University of California, Riverside, CA 92521, USA

Email: bhanu@cris.ucr.edu

Yingqiang Lin: Center for Research in Intelligent Systems, University of California, Riverside, CA 92521, USA

Email: yqlin@cris.ucr.edu

16

Discovering of classification rules from hyperspectral images

Arnaud Quirin and Jerzy Korczak

16.1. Introduction

The emergence and the improvement of remote sensing, aircraft simulation, airborne and spaceborne sensor systems, as well as other kinds of such survey technologies have considerably enhanced our means to explore and to collect data. However, this rapid increase in data results in more time and cost for storage as well as for the data analysis. At the same time, a lot of useless information can hide valuable information. These observations force classification systems to focus on elaborated and sophisticated algorithms to overcome this rapid data growth.

For many years, the design of efficient and robust image classification algorithms has been the most important issue addressed by remote sensing image users. Strong effort has been devoted to elaborate new classification algorithms and improve techniques used to classify remote sensing images using traditional and statistical techniques such as support vector machines [7] or neural networks [16, 21, 22]. But, to our knowledge, relatively few researchers in the evolutionary community have considered how classification rules might be discovered from raw and expertly classified images. Only some works have been done using genetic programming approach [15, 28], but no papers have been published about the effectiveness of learning classifier systems in this field (this review concerns the main conference on learning classifier systems, *IWLCS* (International Workshop on Learning Classifier Systems) from 1992 to 2005). To discover classification rules, the unique source of information is a remote sensing image and its corresponding identification is furnished by an expert. Generally the images, registered by various satellites (e.g., SPOT, CASI, Quick Bird), contain voluminous data. Sometimes they are very noisy due to the presence of various details in a high spatial resolution or unfavorable atmospheric conditions at the time the images were acquired. These data can embrace different cameras having various spectral and spatial resolutions [18, 24, 32]. This chapter presents the potential contribution of evolutionary-based techniques to discover the rules. Learning classifier

systems can produce accurate, robust, and maximally specific classification rules able to deal with the noisy artifacts contained in remote sensing images.

The aim of this chapter is to describe a process of design and validation of evolutionary classifiers applied to remote sensing images. This system is data driven because it generates classification rules able to adapt themselves according to the available data and expertises, and it distributes the quantity of rules in an optimal way to describe each class according to the complexity of the data. In general, classification rules are discovered from the established classifier system [12, 35]. In remote sensing, the initial population of classifiers is randomly created from images and given classes, and then evolved by a genetic algorithm until the acceptable solution is found.

In remote sensing literature, several classification approaches are presented, namely, the following.

- (i) Pixel-by-pixel: each image pixel is analyzed independently of the others according to its spectral characteristic [14].
- (ii) Zone-by-zone: before classification, the pixels are aggregated into zones, the algorithms detect the borders of the zones, delimit them by their texture, their repetitive motives [19].
- (iii) By object: this is the highest level of recognition, the algorithms classify semantic objects, the algorithms detect their forms, geometrical properties, spatio-temporal relations using the background domain knowledge [17].

Our approach is based on spectral data of pixels; therefore, discovered classification rules are only able to find spectral classes rather than semantic ones. This spectral component of class description is essential to well-recognized thematic classes. It should be noted that the proposed classifier system may be easily adapted to more sophisticated object representations.

To validate our approach, a system called *I see you* (ICU) has been used. In the ICU, we have adapted and extended ideas developed in the well-known classifier systems such as XCS [35], the S-classifiers, and “fuzzy-to-classify system” [25]. We have also been inspired by the works of Riolo [27] on gratification and penalization, and of Richards [26] on the exploration of the space of classifiers. To demonstrate the performance of our classifier system, the ICU has been compared with XCS-R and two popular methods, one based on neural networks and the other based on support vector machines [7, 16]. XCS-R is a system based on XCS, integrating the concept of continuous values [36]. XCS is a learning classifier system, developed by [33, 34], that evolves a rule set online based on prediction accuracy and a niched genetic reproduction [20]. The classification systems have been tested in the framework of the European TIDE project [3] on hyperspectral remote sensing images covering the region on Venice.

The chapter is structured as follows. The basic terms and properties of hyperspectral images are introduced in Section 16.2. Section 16.3 gives general ideas about what is a “good” classifier system in remote sensing. Section 16.4 describes our algorithm ICU. In this section, the main components and the quality measures of the method are explained. Section 16.5 draws the main principles of XCS-R and

the adjustments which were made to make it able to deal with these data. The algorithms are evaluated on remote sensing images covering the region on Venice in the framework of the European TIDE project [3]. The results of the comparison between ICU, XCS-R, and two statistical methods (SVM and NN) are presented in Section 16.6. And finally, Section 16.7 concludes the experimentations and indicates the perspectives of the future research.

16.2. Hyperspectral remote sensing images

A hyperspectral image is a set of two dimensional arrays $I_{X,Y,S}$ where (X, Y) is, respectively, the width and the height of the image and S is the number of spectral channels (or spectral bands). The term *hyperspectral* refers to an image which includes more than 20 spectral bands (similar to those produced by ROSIS and DAIS [2]). Conversely, the term *multispectral* is used in the case of a low number of spectral bands such as CASI and Quick Bird remote sensors [30]. A value $I(x, y, s)$ in this array is the reflectance observed on the pixel location (x, y) at the wavelength corresponding to the spectral channel s . A reflectance value corresponds to the intensity of the response obtained from the ground. The input space of a classification problem can be viewed as an ordered vector of real numbers. For each pixel, the spectral signature of this pixel was used. Figure 16.1 shows the spectrum of reflectance of a pixel from a hyperspectral sensor (type MIVIS). Each spectral channel has roughly 10 nanometers in width.

The image data is very voluminous; typically 20 to 200 spectral channels in an image and their size can reach 8000×12000 pixels. Sometimes, half of bands is noisy because of sensor defects or atmospheric absorption of reflectance value in low wavelengths (see Figure 16.2).

To illustrate our approach, two kinds of hyperspectral images have been used. CASI data. Generated by the airborne spectrometer CASI (compact airborne spectrographic imager), with a size of 1175×673 pixels, 288 spectral channels (412–957 nm), and a high resolution (1.3 m). This image has been pre-processed by geometric correction and warping (specifically, first order polynomial warping and nearest neighbor resampling).

MIVIS data. Generated by the MIVIS sensor (multispectral infrared and visible imaging spectrometer) embarked on a satellite, with a size of 397×171 pixels, 92 to 102 spectral channels (433–2478 nm), and a high resolution (2.6 m). Same preprocessing has been applied as before.

Learning and testing were applied on subsets of these images. Subsets contained 142×99 points, and only 1540 points were validated by human ground truthing (expertise ratio: 11%). Then, according to the validation strategies used (hold-one-out, cross-validation), testing sets represent 20% to 50% of the original validated image points. Figure 16.3 shows Quick Bird data for the Lagoon of Venice and the corresponding set of validated points. The rectangle is the area in which all validated points are situated. It should be noticed that the proportion of these points to the whole image is very small—about 0.01%.

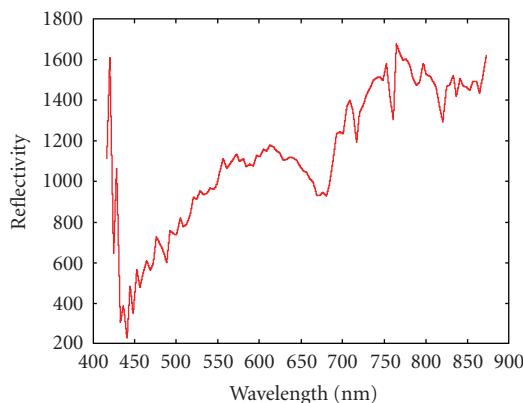


FIGURE 16.1. Spectrum of reflectance observed for a pixel from the hyperspectral sensor MIVIS.

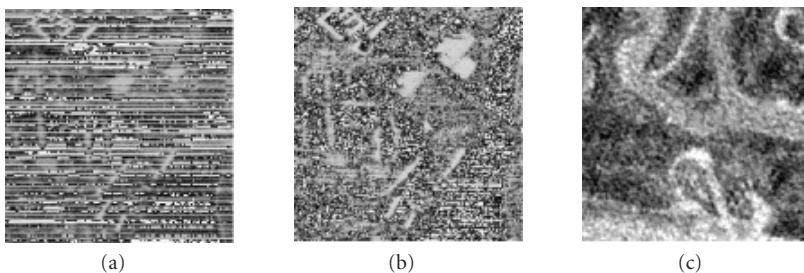


FIGURE 16.2. Typical noisy channels: Strasbourg (band number 42 and number 59 from DAIS) and Lagoon of Venice (band 45 from DAIS), respectively.

16.3. Definition of a system of classifiers in remote sensing

Generally speaking, a system of classifiers integrates symbolic learning and evolution-based computing. Classification rules are symbolic expressions and describe conditions to be held and actions to be taken if the conditions are satisfied. Quality of the rules is evaluated according to their classification performance. Here, we must underscore the fact that the rules are not introduced by a programmer or by an expert.

A system of classifiers is called evolutionary if it is able to adapt itself to the environment. This means that it can modify its knowledge and its behavior according to the situation. For example, in remote sensing, the size of the classes may evolve in one of two ways: (1) if, after an initial classification, there remain nonclassified pixels, or (2) if there are pixels belonging to several classes (mixed pixels). When a classifier integrates one of these pixels to one or another class, it is necessary to dynamically adapt classification rules. In this way, certain rules that treat only the simple cases (not mixed pixels) will become useless, and new rules are necessarily created for other cases.

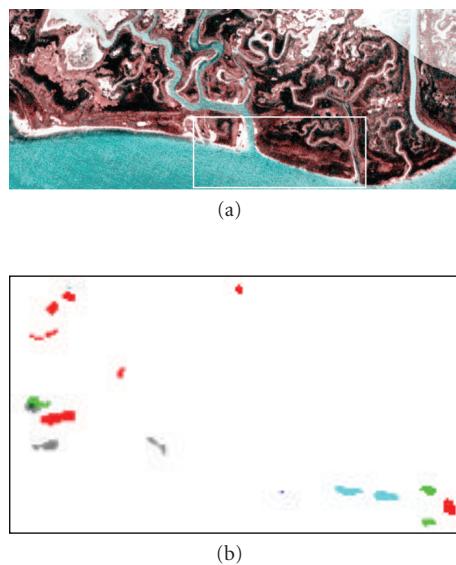


FIGURE 16.3. Quick Bird data of the Lagoon of Venice and the corresponding set of validated points (ground truthing).

From a functional point of view, a classifier can be defined as a rule representing a piece of knowledge about a class, and may be a conditional expression, such as *if* “conditions” *then* “action”. In the early classifier systems [35], each part of a rule was a binary message, encoding elementary information such as a value, colour, form, shape, and so forth. The “conditions” part described an entry message in the system, corresponding to conditions that must be fulfilled in order to activate this rule. The “action” part defined the action to be carried out when the appropriate conditions were satisfied. This binary encoding scheme is not well adapted to image classification rules.

One of the reasons for this is based on the domain of spectral values that may be assigned to a pixel (from 0 to 255 for 8-bit pixels, or from 0 to 65000 for 16-bit pixels). Of course, binary encoding of rule conditions is possible but the rules would be difficult to understand. Instead, we assert that the evolved rules must be rapidly evaluated and easy to interpret by any user. As a result, condition representation using the concept of an interval could be fully adequate for remote sensing image classification. In terms of machine learning, the rules have to be maximally specific generalizations, meaning that they have to cover the maximum pixels belonging to a given class and the minimum pixels belonging to other classes.

Before rule specification is explained, recall that a pixel is encoded as a spectral vector, defining a value of reflectance for the n bands of the remote sensing image:

$$\langle \text{pixel} \rangle := [b_1, b_2, \dots, b_n]. \quad (16.1)$$

In our system, the condition for any rule is built on the concept of spectral intervals defining a given band, corresponding to a given class. Such intervals are a pair of integer numbers, between 0 and the maximum possible value for a pixel of a given band (i.e., 65536 for the pixels defined on 16 bits). This solution allows to partition the space of the spectral values in two ranges: the first containing the pixel values which corresponds to a given class, and the second containing the remainder.

To precisely specify the class definition, a set of intervals is defined for each band of the remote sensing image. Taking into consideration all bands, the condition part is defined as a set of hyper-rectangles in an \mathbb{R}^n space:

$$\text{< condition >} := \bigwedge_{i=1}^n \bigvee_{j=1}^k m_i^j \leq b_i \leq M_i^j, \quad (16.2)$$

where m_i^j and M_i^j denote, respectively, the minimal and maximum reflectance values allowed for a pixel belonging to a class C for band i . k is a fixed parameter which defines the maximum number of disjunctions allowed.

The intervals $[m_i^j; M_i^j]$ are not necessarily disjunctive. By experiments, we have found that if we allow the genetic algorithm to create nondisjunctive intervals, instead of merging them, the results of genetic operators are more interesting. We have noticed that merging intervals significantly diminishes the number of intervals, and at the same time reduces the possibilities to create more efficient rules. The example below illustrates a concept of interval merging: $E = [11; 105]$ or $[138; 209]$ or $[93; 208]$ corresponds after merge operation to $E = [11; 209]$.

To satisfy a rule, a pixel has to match at least one spectral interval for each band. Logically speaking, to associate a pixel to a class, its values have to satisfy the conjunction of disjunctions of intervals that define a condition part of the classification rule. Figure 16.4 illustrates an example of matching of two pixels against the spectral class. The left figure graphically shows the spectral intervals of the class defined by a given rule. The next two diagrams show spectral signatures of two pixels: the first matches the rule, but the second does not. Hence, only the first pixel of this example may be considered to be an instance of the class.

This representation of the rule has been chosen mainly because of its simplicity, compactness, and uniform encoding of spectral constraints. During experimentation, this representation has demonstrated rapid execution of genetic operators and efficient computing. Of course, one may specify more complex structures using spatial properties of the pixel, with respect to the pixel neighborhood. Also, one may include features resulting from thematic indices or mathematical operators applied to pixel environment. We may also apply a genetic programming to identify new characteristics. These semantically extended formalisms are interesting, however, they not only require more sophisticated genetic operators, but also more powerful computers to perform the calculation in an acceptable amount of time.

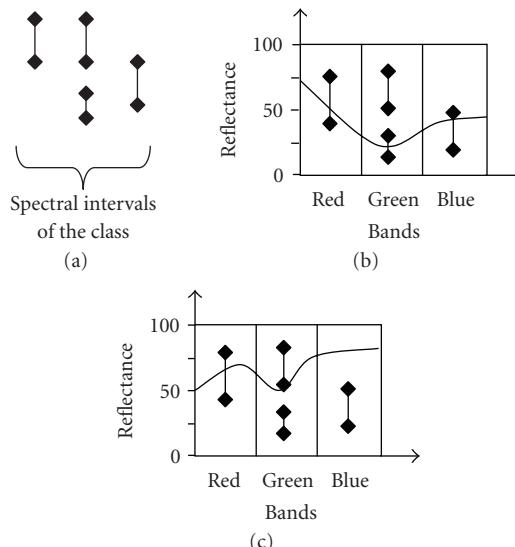


FIGURE 16.4. Matching spectral bands and spectral signature of pixels. For instance, a real-image sample for the rule could be (see the figure): “if $40 < R < 80$ AND $(20 < G < 30$ OR $55 < G < 85)$ AND $25 < B < 50$ then class X,” where R , G , and B are the channels of a three-band captor.

16.4. From the rule creation to the evolution with ICU

16.4.1. Genetic algorithm

To discover a classification rule, ICU uses a Michigan-like learning classifier system representation, in which each rule is encoded by one individual [24]. In order to efficiently develop the classification rules, a genetic algorithm initializes interval values according to spectral limits of the classes designated by an expert, for valid zones of the remote sensing image. Initial classification rules are created based on the extreme maximum and minimum values for defined spectral intervals, taking into account every class. It should be noted that by this initialization, rule searching is considerably reduced, and initial intervals are very close to the final solution. During this process, the initial spectral limits are slightly perturbed by adding a random value to lower and upper spectral limits. Hence, the initial population of classification rules is quite diversified.

This initial pool of classifiers is evolved from a genetic algorithm. Our system searches for a best classifier for each class, independently. A major reason for choosing this procedure is the efficiency of computations; that is, the process of rule discovery is not perturbed by other rules.

The quality of classification rules is based on a comparison of these results with the image classified by an expert. If pixels covered by the classifier perfectly overlap those indicated by an expert, then the system assigns the highest quality value to the classifier; otherwise, in the case of some mismatching, the quality factor is reduced (between 0 and 1). An associated fitness function will be detailed

```

{R is a classification rule and  $P_0$ ,  $P_1$  and  $P_2$  populations of
classification rules}
R := INITIAL_RULE(images)           // Creation of a rule
according to spectral extremes
 $P_0 := \text{INITIALIZATION}(R)$            // Random perturbation of rules
EVALUATION( $P_0$ )                   // Calculation of the fitness function
for each rule
while TERMINATION_CRITERION( $P_0$ ) = false do
     $P_1 := \text{SELECTION\_X}(P_0)$            // Selection for crossover
     $P_1 := \text{Crossover}(P_1) \cup \text{COPY}(P_0)$ 
     $P_2 := \text{SELECTION\_MUT}(P_1)$            // Selection for mutation
     $P_2 := \text{MUTATION}(P_2) \cup \text{COPY}(P_1)$ 
    EVALUATION( $P_2$ )
     $P_0 := \text{REPLACEMENT}(P_0, P_2)$            // New generation of rules
end while
Result: R, the classification rule for a given class

```

ALGORITHM 16.1. Process of rule discovery.

TABLE 16.1. Characteristics of the evaluation function.

		Image classified by the classifier R (I_{rule})	
		No. of pixels activating R	... nonactivating R
Pixel classified by the expert E (I_{expert})	True	$p_{\text{exp}}^{\text{ul}}$	$p_{\text{exp}}^{\text{ul}}$
	False	$p_{\text{exp}}^{\text{ul}}$	$p_{\text{exp}}^{\text{ul}}$

in the next section. During the evolution process, the rules are selected for the crossover according to the quality for a given class. The process of rule evolution, that we have used, is defined in Algorithm 16.1.

As mentioned before, this algorithm must be designed to run independently for each class. This allows for obtaining classifiers according to user requirements without the necessity of carrying out computations for all classes with the same level of quality. This also allows for the maintenance of previously generated classifiers, as well as for the introduction of new ones. Further, the user may define a hierarchy of classes and specialize some classifiers while respecting newly created subclasses with different levels of classification quality.

16.4.2. The evaluation function

The evaluation function serves to differentiate the quality of generated rules and guide genetic evolution. Usually, this function depends strongly on application domain. In our work, evaluation is based on the classification obtained by the classifier (I_{rule}) and the expertly given classification (I_{expert}). Table 16.1 defines the values necessary to compute the evaluation function.

In a given system, the evaluation function is computed as a balanced score between sensitivity and specificity, two popular quality measures in remote sensing:

$$N_{\text{final}} = \alpha N_{\text{class}} + (1 - \alpha) \overline{N_{\text{class}}}, \quad (16.3)$$

where $N_{\text{class}} = P_{\text{exp}}^{\text{rul}} / (P_{\text{exp}}^{\text{rul}} + P_{\text{exp}}^{\overline{\text{rul}}})$ is the sensitivity and $\overline{N_{\text{class}}} = P_{\text{exp}}^{\overline{\text{rul}}} / (P_{\text{exp}}^{\text{rul}} + P_{\text{exp}}^{\overline{\text{rul}}})$ is the specificity. α is called the adjusting coefficient, which is used for certain classes that are under or over represented. By default, the value of this coefficient is equal to 0.5.

The proposed function has a number of advantages; it is independent of the pixel processing sequence, invariant of the size of classes, and efficient for class discovery with a highly variable number of pixels.

The evolution process converges according to some statistical criteria indicating if the current classifier is near to a global optimum or if the population of rules will not evolve anymore. The termination criterion of the algorithm leans on the statistics of classifier quality evolution. In our system, we take into consideration not only the evolution of quality of the best and the average classifiers, but also the minimum acceptable quality defined by a user and a maximal number of generations to run. If one of these criteria is satisfied, then the process is stopped.

The most difficult determination to make is whether the quality of a classifier is not continuing to evolve. To detect stabilization of the quality measure, we have based our heuristics on statistics regarding quality evolution of the best classifier. For example, let Q_k be the quality of the best classifier obtained during the last k generations, and Q_o be the quality of the best classifier of the current generation. The algorithm is interrupted if the following equation is satisfied:

$$\left| \frac{\sum_{k=1}^P Q_k}{P} - Q_o \right| \leq E, \quad (16.4)$$

where P represents the maximum period of quality stabilization, and E is a maximal variation of this stabilization compared with the current quality.

It is important to have an initial population of classifiers within the vicinity of the solution to be found. Two algorithms have been proposed allowing for the generation of a diversified pool of classifiers close to the expert hidden classification rule. The first, called *GenMinMax*, creates maximum intervals covering the expert rule, and the second algorithm, called *GenSpectro*, integrates the spectral distribution density and interval partitioning [24].

16.4.3. Genetic operators

One of the most important tasks while designing a genetic algorithm is to invent operators that will create new potential solutions. All of our operators have been

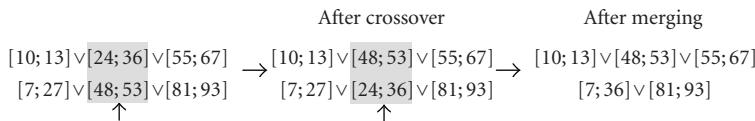


FIGURE 16.5. Interval merging after crossover operation. With a real image, the crossover consists to exchange the tests corresponding to given parts of the spectrum of a sample. Interval merging does not change the mathematical interpretation of the test.

specialized on classifier representation, and they have been validated on remote sensing images. With respect to software engineering, the genetic algorithm has been structured into layers corresponding to genetic operations (e.g., selection, mutation, crossover, and replacement). The system is viewed as a collection of layers with data passed from layer to layer. Layer execution follows from one to another, and genetic operations are invoked in the same sequence. This modular approach makes program maintenance and future extensions much easier.

Selection of classifiers. In general, selection is the operation of allocating reproductive opportunities to each classifier. The reproductive force of a classifier is expressed by a fitness function that measures the relative quality of a classifier by comparing it to other classifiers in the population. There are many methods for selecting a classifier [6]. In our system, the selection operator is applied in the following cases: (1) choice of the classifier to be reproduced for the crossover or the mutation; (2) choice of the classifier to be copied during the reproduction (see the first chapter); and (3) choice of the classifiers to be kept or eliminated during the replacement.

Selection strategies are well known: the roulette wheel, ranking, elitism, random selection, the so-called tournament, and eugenic selection. Our experiments have shown that roulette wheel selection is most advantageous for the reproductive phase, but the tournament strategy with elitism is best for the generational replacement scheme.

Crossover of classifiers. In our case, the crossover operator needs two parents, and cuts their chromosome at some randomly chosen positions to produce two offspring. The two new classifiers inherit some rule conditions from each parent-classifier. Then, each result of the crossover process has to be validated. Validation of the various rule attributes (border limits violation, overpasses, etc.) is carried out by a process of interval merging, as shown in Figure 16.5.

However, merging not only decreases the number of intervals in the rules, but also generates some information loss. In fact, in order to avoid a premature convergence of rules, it is generally important to preserve for the following generation two distinct intervals instead of a single aggregated one. On the other hand, it is interesting to note that the positive or negative effects of an interval on the quality of the rule can be related to other intervals encoded in the classification rule. By checking the quality of the obtained rules, we finally discovered that *not merging*

is a better solution if the user is not concerned by the computing time because it requires more iterations.

Mutation of classifiers. The mutation processes a single classification rule and it creates another rule with altered condition structure or variables. In our system, the mutation operator may be applied on three levels: band level, interval level, and border level.

Band mutation consists of a deletion of spectral bandwidth in a chosen classification rule. Its interest is twofold; firstly, the band mutation type allows for simplification and generalization of a rule; secondly, it allows for the elimination of noisy bands that frequently appear in hyper spectral images. The existence of noisy bands significantly perturbs the learning process, as well as the process of evolution convergence.

Interval mutation allows for a chosen band to add, eliminate, or cut an interval into two spectral ranges. In case of addition, the new rule is completed by a new interval centered randomly with a user-defined width. The cutting of an interval is done by random selection of a cutting point within the interval (e.g., the cutting of [10;100] can generate two intervals: [10; 15] and [15; 100]). Mutation such as this allows for breakage of continuous spectral ranges. And, this allows for the definition of a spectral tube in which spectral values of the pixels can be assigned to a given class.

Finally, border mutation modifies both boundaries of an interval. This mutation refines the idea of targeting spectral tubes carried out by the other types of mutation. It is worthwhile to note that the mutated rules are systematically validated.

In our system, mutation operators are dynamically adapted by tuning (i.e., we test different parameters for the same training set and we elect the best choice by looking at a testing set). Adjustment is related to the probability of each mutation operator according to its current effectiveness.

Generational replacement. In our system, the two strategies presented in the introduction, that are the (μ, λ) and the $(\mu + \lambda)$ strategies can be applied. According to Algorithm 16.1, the new generation of classifiers is created from a population of parents (P_0 of μ individuals) and their children after the crossover and the mutation operators (P_2 of λ individuals). Depending on a parameter set by the user, the **REPLACEMENT()** operator can apply the (μ, λ) or the $(\mu + \lambda)$ strategy. Best results were obtained with the (μ, λ) strategy.

There exist other replacement strategies integrating for instance heuristics where the best individual of the previous population replaces the worst one of the current population or heuristics where the new individuals having a performance higher than a certain threshold are inserted. However, both of these strategies present the risk of having the same individuals remain in the population. This is not necessarily a problem except in the case of a weak genetic pool in which some individuals of average performances that would profit from immunity.

16.5. From the rule creation to the evolution with XCS-R

XCS and XCS-R are learning classifier systems (LCS). LCSs are algorithms for symbolic learning. The main difference between LCSs and other learning techniques like neural networks and genetic programming is the aptitude for collaborative learning. The objective is to create a chain containing a given number of classifiers, so that the whole chain will be the complete solution of the problem. The information will be transmitted from a link to another, using messages to interpret.

XCS evolves a set of rules, the so-called *population of classifiers*. This method is presented for comparison with ICU. Rules are evolved by the means of a genetic algorithm. A classifier usually consists of a condition and an action part. The condition part specifies when the classifier is applicable and the action part specifies which action, or classification, to execute. In contrast to the original LCSs, the fitness in the XCS classifier system, introduced by Wilson [33], is based on the *accuracy* of reward predictions rather than on the reward predictions themselves. Thus, XCS is meant to evolve not only a representation of an optimal behavioral strategy, or classification, but rather to evolve a representation of a complete payoff map of the problem, that is, XCS is designed to evolve a representation of the expected payoff in each possible situation-action combination.

Since our system is confronted with real-valued data in this study, we apply the real-valued extension of XCS (XCS-R) introduced in [36]. Recently, several studies have been reported that show that XCS performs comparably well to several other typical classification algorithms in many standard data mining problems [4, 5, 13].

This section provides a short introduction to the XCS classifier system. For a more detailed introduction to XCS and XCS-R, the interested reader is referred to the original paper [33], the real-valued extension [36], and the algorithmic description [9].

16.5.1. Overview of the algorithm

As mentioned, XCS evolves a population [P] of rules, or classifiers. Each classifier in XCS consists of five main components.

- (1) The condition part specifies the subspace of the input space in which the classifier is applicable, or matches. In our real-valued problem, a condition specifies a conjunction of intervals, one for each attribute. If the current problem instance lies within all specified intervals, the classifier matches.
- (2) The action part specifies the advocated action, or classification.
- (3) The payoff prediction estimates the average payoff encountered after executing action A in the situations in which the condition part matches.
- (4) The prediction error estimates the average deviation, or error, of the payoff prediction.
- (5) The fitness reflects the scaled average relative accuracy of the classifier with respect to other overlapping classifiers.

Learning usually starts with an empty population. Given current input, the set of all classifiers in $[P]$ whose conditions match the input is called the match set $[M]$. If some action is not represented in $[M]$, a covering mechanism is applied. Covering creates classifiers that match the current input and specify the not-covered actions.¹ Given a match set, XCS can estimate the payoff for each possible action forming a prediction array $P(A)$. Essentially, $P(A)$ reflects the fitness-weighted average of all reward prediction estimates of the classifiers in $[M]$ that advocate classification A . The payoff predictions determine the appropriate classification. During learning, XCS chooses actions randomly. During testing, the action a_{\max} with the highest value $P(a_{\max})$ is chosen.

16.5.2. Reinforcement and discovery components

XCS iteratively updates its population of classifiers with respect to the successive problem instances. After the classification is selected by the means of the prediction array and applied to the problem, scalar feedback is received. In a classification problem, classifier parameters are updated with respect to the immediate feedback in the current action set $[A]$, which comprises all classifiers in $[M]$ that advocate the chosen classification A . After rule evaluation and possible genetic algorithm (GA) invocation, the next iteration starts.

The aforementioned covering mechanism ensures that all actions in a particular problem instance are represented by at least one classifier. Each attribute of the new classifier condition is initialized using parameter cover-rand that specifies the maximal interval the condition comprises in an attribute. XCS applies a GA for rule evolution. A GA is invoked if the average time since the last GA application upon the classifiers in $[A]$ exceeds a threshold. The GA selects two parental classifiers using set-size relative tournament selection [8]. Two offspring are generated reproducing the parents and applying crossover (uniform crossover) and mutation. Parents stay in the population competing with their offspring. In the insertion process, subsumption deletion may be applied [34] to stress generalization. Due to the possible strong effects of action-set subsumption, we only apply GA subsumption, which searches for an accurate, more general classifier that may subsume the offspring. If such a more general classifier is found, the offspring is discarded and the numerosity [34] of the subsumer is increased. The population of classifiers $[P]$ is of fixed size N . Excess classifiers are deleted from $[P]$ with probability proportional to an estimate of the size of the action sets that the classifiers occur in. If the classifier is sufficiently experienced and its fitness F is significantly lower than the average fitness of classifiers in $[P]$, its deletion probability is further increased.

¹ICU benefits from an initialization creating directly good initial individuals by looking at the whole training set. However, XCS has this covering mechanism that could create good individual but by looking only at one sample of data. It is difficult to state which technique is better, because this strongly depends on the implicit patterns contained in the data. The difference of the accuracies of the two algorithms observed in the case study is caused by the preprocessing of the data (if applied) and the initialization phase. More theoretical studies are required to predict, in the general case, which method is more suitable.

16.5.3. Rules selection

As in ICU, the same pixel may activate several rules coding for different classes. This behavior can be constrained for solving problems as *one-to-one* classification (one pixel always corresponds to one class, contrary to the *one-to-N* classification, in which one pixel may correspond to several classes). Two methods (*MaxConfident* and *ScoredConfident*) were tested, asking the pool to give a unique class for each pixel.

- (1) In the first one, only the rules which have a correct self-confidence are considered, in other terms, payoff prediction should be greater than a given threshold (fixed in our experiments to the half of the maximum value of payoff prediction for the current problem). Then the most frequent class obtained from rules which had matched the pixel is returned.
- (2) In the second one, all the rules which have matched the pixel are considered. For a class c , a score is computed as follows for each rule r :

$$S_r = \frac{\sum P_r * F_r}{\sum F_r}, \quad (16.5)$$

where P_r is the prediction payoff of the rule r and F_r is its fitness. The action of the rule with the best S_r is returned.

These two methods have been experimented for different subsets of CASI data and for a pretreated expertise set as follows: if more than one class was affected to the same pixel, only the dominant class was retained according to the percentage of its concentration given by the expert. Here, contextual classification (i.e., looking at the neighbor pixels) can help to overcome this kind of ambiguity. Many cases of ambiguities occur when two classes have the same spectra whereas they can be discriminated by the context (e.g., water and shadows have relatively low reflectance and they can be distinguished by looking at the classes located around).

16.6. Case studies

16.6.1. Consideration with the data

The data available for this study was acquired during a field campaign at the end of September 2002, for the European project TIDE (Tidal Inlets Dynamics and Environment, [3]). During this project, data was obtained from satellite or aircraft, at different scales and resolutions, providing a *multilevel view* of the ground [29]. A multilevel remote sensing is a useful technique to monitor large areas: a global view of the area can be identified by the satellite image, while the checking of a particular area or a classification needs aerial imagery. The expertise used in the following supervised classification are based on a costly mean, ground truthing (the characterization of all points are made by hand by a human expert), and on expert validation provided by the examination of different levels of the data. Due to the existence of such different sources, the expert validation of points is not

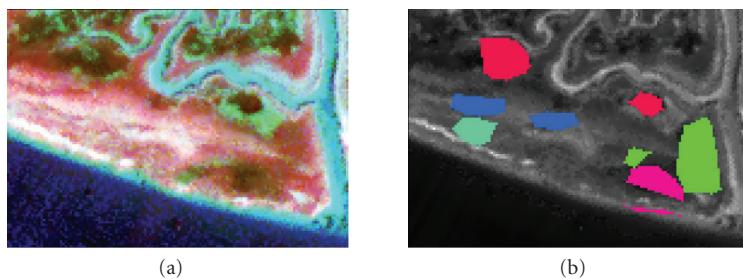


FIGURE 16.6. Reference and expert data (CASI, San Felice).

the easiest way but it is relatively safe and relevant. Some assumptions were made about the data [31].

- (i) The reference data in the learning base are truly representative of the sought classes.
- (ii) The reference data and the expert data are perfectly synchronized by geo-referencing.
- (iii) There is no error in the reference data (incorrect or missing class assignments, change in the vegetation boundaries between the time of imaging and the time of field verification, positional errors, etc.).

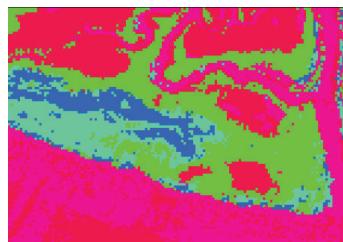
Considering the expertise, two issues have to be addressed. The first, the complexity of the analyzed environment, particularly the effect of partial volume (data includes *pixels* of nonpure classes), requires that the expert selects several classes for one point (*multilabeling*). A typical supervised method handles only one class attribute. Thus, in the case of multilabeling, the dominant class is kept or the point is dropped. The second mainly for cost reasons, the human expert can only label very few points. The identification of the boundary of each class and the construction of convex hulls are mandatory to include more and interior pixels in the expertise. The points can represent exactly the class, a part of the class, or sometimes a corner of a different kind of vegetation cross a hull supposed to be a whole class. Extra knowledge may be needed to select the correct points.

In this chapter, a remote sensing image of San Felice (Lagoon of Venice) has been chosen (Figure 16.6). This image contains multispectral data (CASI 15 bands), with 142×99 pixels, 16 bits per pixel, and 1.3 m terrain resolution [24]. The case study considers a typical problem of classification for rural zones, with only five requested classes but including a high percentage of mixed pixels. Learning was carried out on 50% of the 1540 points, and then validation was performed on the whole data.

16.6.2. ICU classification

The image and statistics presented in Figure 16.7 summarize our experiments.

The parameters (described in Section 16.4.3) for ICU are as follows: $P_{cross} = 0.7$, $P_{mut} = 0.15$, $P_{mut,band} = 0.3$, $P_{mut,interval} = 0.2$, $P_{mut,border} = 0.4$,



(a)

		Classification by the expert					
		SAR2	LISR	SPA2	SPA1	LIM1	Q _{ppa}
By the classifiers	SAR2	222	0	0	0	5	98%
	LISR	59	149	0	0	21	65%
	SPA2	0	0	382	162	8	69%
	SPA1	0	0	0	28	0	100%
	LIM1	0	0	3	33	468	93%
	Q _{sens}	79%	100%	99%	13%	93%	

(b)

FIGURE 16.7. Classified image and confusion matrix for ICU.

300 individuals, 2000 to 5000 generations and selection by rank for the crossover and the mutation operators.

It should be mentioned that validation points do not concern any water. Water can be classified quickly using standard statistical tools, even with unsupervised ones, and are beyond the interest of the experts. ICU classifies these pixels in the most approximate class (SPA1), but quality of these areas are not relevant (at the south and the right border). SPA2 is a pure class of Spartina Maritima and SPA1 contains Spartina in a nondominant way. The only point where ICU disagrees with the expertise is the small spot at the right lower corner (SPA1 instead of SPA2, see Figure 16.7). The very narrow resemblance of the spectra of these two classes explains that. Classification is quite globally correct (κ -index of 0.81, average accuracy of 81.1%). Figure 16.8 shows the map of overlaps, designed to test overlapping rules. The whiter is a pixel, the more rules are activated. If no rule can be used, the pixel is shown in red (in fact in the color “0” depending on your printout). The map of overlaps is automatically generated, it shows pixel classes and the degree of mixing, without any external knowledge. The image below demonstrates, for instance, that no knowledge about water evolved.

16.6.3. XCS classification

The classification with XCS shows quite good results. The image and statistics presented in Figure 16.9 illustrate our experiments.

The same remark should be made about the color of what one believes being a water class, which actually is not relevant. ICU and XCS find the same classes for the same points, including the border of the salt marsh, which contains a lot of mixels (a composition of several species on the same pixel). XCS has spent many

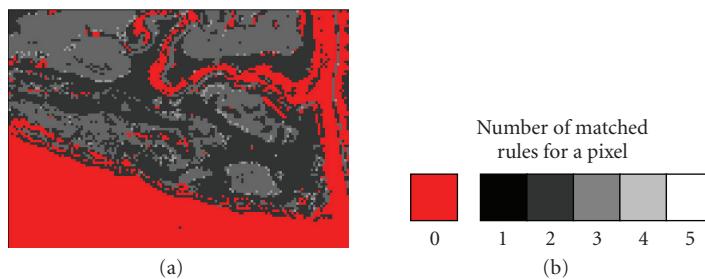
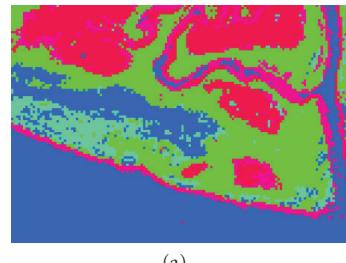


FIGURE 16.8. Map of the overlapping zones, produced only by ICU.



		Classification by the expert					
		SAR2	LISR	SPA2	SPA1	LIM1	Q _{ppa}
By the classifiers	SAR2	266	12	0	0	3	95%
	LISR	11	133	0	0	2	91%
	SPA2	0	0	353	83	1	81%
	SPA1	0	0	27	106	3	78%
	LIM1	4	4	5	34	493	91%
	Q _{sens}	95%	89%	92%	48%	98%	

(b)

FIGURE 16.9. Classified image and confusion matrix for XCS.

classifiers to separate SPA1 and SPA2 correctly. Nearly one classifier was used to describe each pixel. Nevertheless, a better global accuracy was obtained (κ -index of 0.88, average accuracy of 87.7%). XCS is then better than ICU with a less accurate initialization procedure, because the pattern of this data could be better learned with local initializations (covering operator) than using a global initialization (*GenMinMax* or *GenSpectro* of ICU). But global initialization is more fast. The typical learning time with ICU on this dataset is about 2–3 minutes, against 4–7 minutes on a 3 GHz CPU.

16.6.4. Comparison between ICU and statistical classifiers

To attest that performance of ICU is comparable to other algorithms known to be really efficient in this domain [7, 16, 21, 22], this algorithm was tested on two data sets. These data correspond to two images of San Felice (Lagoon of Venice), with different spectral and spatial resolutions, namely, CASI sensor (date: 2002, 15

TABLE 16.2. Results obtained for the two images.

	Accuracy for CASI (κ -index)	Accuracy for MIVIS (κ -index)
ICU	0.990 (0.99)	0.884 (0.88)
SVM	0.974 (0.97)	0.978 (0.98)
Neural network	0.893 (0.89)	0.878 (0.88)

bands, 754×293 pixels, resolution 1.3 m^2 , 6 classes) and the hyperspectral MIVIS sensor (date: 2003, 20 bands, 396×170 pixels, resolution 2.6 m^2 , 7 classes).

The results for the testing set (50% of all of the data) are presented in Table 16.2. Accuracy is the mean of the diagonal of the confusion matrix and the κ -index characterizes a correctly distributed confusion matrix (1 on the diagonal, 0 everywhere else).

For ICU, the parameters were the same as those described in Section 16.6.2. For the algorithm based on neural networks, a learning rate of 0.1, 100000 iterations, 1 hidden layer of 7 to 15 neurons, an incremental method for the learning, and a symmetrical sigmoid activation function were chosen. The exit layers represent the expert continuous values and there are as many exit neurons as there are classes to be learned. The free library in C, named the fast artificial neural network library (FANN, [1]), was used. The chosen neural network topology was simple but efficient. For the algorithm based on support vector machine (SVM), the RBF kernel was used; and the parameters C and γ were discovered for each data set using free software and a step-by-step optimization algorithm in Python presented on the site of the LIBSVM [10].

ICU has shown better performance than SVM and NN for the CASI image and better than NN for the MIVIS image. The image of the MIVIS sensor was most difficult to analyze for several reasons: some additional bands; less pixels in the image, thus less samples in the training set (each time the whole training set consisted of less than 2% of the image); the first bands of MIVIS are slightly more disturbed than for CASI; and finally some classes of vegetation were no longer present on the saltmarsh in 2004, which caused a drop in the quality of the training data. When the values in the data set are a kind of composition of several pure classes (*pixels*), a system based on disjunctions like ICU proves to be more useful. Moreover, this kind of classifiers would make it possible to expose the contents of the rules to a human expert contrary, for example, to a neural network (often treated as a *black box*).

To illustrate the simplicity of the formalism of the rules discovered by our algorithm, an example of a rule which classifies the instances of one of the class follows:

$$\begin{aligned}
 & (435 \leq B_0 \leq 1647) \wedge \dots \wedge ((365 \leq B_{74} \leq 4023) \vee (15643 \leq B_{74} \leq 48409)) \\
 & \wedge \dots \wedge (668 \leq B_{79} \leq 4413) \Rightarrow [\text{CLASS 4}],
 \end{aligned} \tag{16.6}$$

where B_i is the reflectance value for the band i of the considered pixel. The typical learning time with NN on this dataset is about 1 minute, against 10–15 minutes for the step-by-step optimization of SVM on a 3 GHz CPU.

16.6.5. Summary of experiments

The three case studies have demonstrated the high capacity of the evolution-based classifiers to interpret and classify heterogeneous and complex images (e.g., high dimension in (X, Y, S) , large number of bands and noisy data that generate a computational complexity of $O(n^3)$). The quality of classification has been shown very high even in cases of high number of noisy bands and mixed pixels. It must be noted that the quality of learning is highly related to the quality of the classified image used for rule discovery. The discovered classification rules have been in general simple and easy to interpret by remote sensing experts. They are also mutually exclusive and maximally specific. Nevertheless, the learning time was relatively long due to the large image size and the chosen parameters. Classified images by the discovered rules have shown that the evolution-based classifier is able to faithfully reproduce the human expertise and algorithms already in use in this domain.

16.7. Conclusion and perspectives

This chapter has detailed the evolution-based classifier systems applied to remote sensing images. The systems have been able to discover a set of “*if ... then ... class*” classification rules using the fitness function based on image classification quality. These rules have been proven to be robust and simple to understand by the user. The accuracy of the expert has been improved and the rules have been demonstrated sufficiently generic for reusing them on other parts of satellite images. Hence, the classifier systems can be considered of great interest when compared to traditional methods of classification.

Taking into consideration image complexity and noisy data, the results of our experiments are very encouraging. Case studies have demonstrated that the obtained classifiers are able to reproduce faithfully the terrain reality. The rules are well adapted to recognize large objects on the image (e.g., sport lands), as well as the smaller ones (e.g., trees, shadows, edges of the buildings). The redundant or noisy bands have been successfully identified by the classifiers. The formalism of rule representation has allowed the modelling of a spectral tube adapted to the granularity of spectral reflectance.

The potential of evolution-based classifiers in remote sensing image classification begins to be explored. Further investigation of the classifiers efficiency are necessary. Currently, we are starting to work on a more powerful representation of rules including spatial knowledge, temporal relations, and hierarchical representation of objects. Contextual classification can help to overcome some ambiguities, as stated in the last section before the case studies, but this technique requires to know in which order the algorithm will classify the neighbor pixels. Future research would be followed in this direction. We are also trying

to optimize system performance, in particular the implementation of the genetic process on a parallel machine and the tuning of its initial parameters. The classifier system developed by this research work, called ICU, and other classification software related to remote sensing are currently available on our web site <http://lsiit.u-strasbg.fr/afd/>. The XCS algorithm can be downloaded from the IlliGAL web site, <http://www-illigal.ge.uiuc.edu>.

Acknowledgments

The authors would like to thank Marco Marani, Enrica Belluco, Monica Camuffo, and Sergio Ferrari from the UNPADU University (Padova, Italia) for the multi-spectral data sets used (images of San Felice and continuous expertise acquired during the TIDE project [3]). The authors are grateful to Mindy Jacobson and Pierre Ganarski, Cdric Wemmert, Christiane Weber, Anne Puissant, and Massimo Menenti from the Louis Pasteur University (Strasbourg, France) for their advices, suggestions, and comments during the early stages of this research, and also to David E. Goldberg and Martin Butz from the IlliGal Laboratory at Urbana-Champaign for providing hints, algorithms, and judicious remarks concerning the XCS algorithm. They also thank their two reviewers for dedicating the time and expertise to improve the presentation and the readability of the chapter.

Bibliography

- [1] “Fast artificial neural network library,” <http://fann.sourceforge.net/>.
- [2] DAIS, *Proceedings of the Final Results Workshop on DAISEX (Digital AIrborne Spectrometer EXperiment)*, ESTEC, Noordwijk, The Netherlands, 2001.
- [3] Tidal Inlets Dynamics Environment, “Research Project Supported by the European Commission under the Fifth Framework Programme,” contract no. EVK3-CT-2001-00064, 2001–2005, <http://www.istitutoveneto.it/tide>.
- [4] E. Bernadó-Mansilla and J. M. Garrell-Guiu, “Accuracy-based learning classifier systems: models, analysis and applications to classification tasks,” *Evolutionary Computation*, vol. 11, no. 3, pp. 209–238, 2003.
- [5] E. Bernadó, X. Llorà, and J. M. Garrel, “Xcs and gale: a comparative study of two learning classifier systems on data mining,” in *The 4th International Workshop on Advances in Learning Classifier Systems (IWLCS '01)*, vol. 2321 of *Lecture Notes in Computer Science*, pp. 115–132, San Francisco, Calif, USA, July 2002.
- [6] T. Bickle and L. Thiele, “A comparison of selection schemes used in genetic algorithms,” TIK-Report 11, Computer Engineering and Communication Networks Lab, Swiss Federal Institute of Technology, Zurich, Germany, 1995.
- [7] M. Brown, S. R. Gunn, and H. G. Lewis, “Support vector machines for optimal classification and spectral unmixing,” *Ecological Modelling*, vol. 120, no. 2-3, pp. 167–179, 1999.
- [8] M. V. Butz, K. Sastry, and D. E. Goldberg, “Tournament selection in XCS,” in *Proceedings of the 5th Genetic and Evolutionary Computation Conference (GECCO '03)*, pp. 1857–1869, Chicago, Ill, USA, July 2003.
- [9] M. V. Butz and S. W. Wilson, “An algorithmic description of XCS,” *Soft Computing*, vol. 6, no. 3-4, pp. 144–153, 2002.
- [10] C.-C. Chang and C.-J. Lin, “LIBSVM: a library for support vector machines,” software, <http://www.csie.ntu.edu.tw/~cjlin/libsvm>, 2001.

- [11] J. M. Daida, D. E. Lund, C. Wolf, et al., "Measuring topography of small-scale waves," in *Proceedings of IEEE International Geoscience and Remote Sensing*, T. Stein, Ed., pp. 1881–1883, IEEE Press, Florence, Italy, 1995.
- [12] K. A. DeJong, "Learning with genetic algorithms: an overview," *Machine Learning*, vol. 3, pp. 121–138, 1988.
- [13] P. W. Dixon, D. W. Corne, and M. J. Oates, "A preliminary investigation of modified XCS as a generic data mining tool," in *The 4th International Workshop on Advances in Learning Classifier Systems (IWLCS '01)*, vol. 2321 of *Lecture Notes in Computer Science*, pp. 133–150, San Francisco, Calif, USA, 2002.
- [14] R. Fjortoft, P. Marthon, A. Lopes, F. Sery, D. Ducrot-Gambart, and E. Cubero-Castan, "Region-based enhancement and analysis of SAR images," in *Proceedings of IEEE International Conference on Image Processing (ICIP '96)*, vol. 3, pp. 879–882, Lausanne, Switzerland, September 1996.
- [15] C. Fonlupt and D. Robilliard, "Genetic programming with dynamic fitness for a remote sensing application," in *Proceedings of the 6th International Conference on Parallel Problem Solving from Nature (PPSN '00)*, pp. 191–200, Paris, France, September 2000.
- [16] X. Jiang, M.-S. Chen, M. T. Manry, M. S. Dawson, and A. K. Fung, "Analysis and optimization of neural networks for remote sensing," *Remote Sensing Reviews*, vol. 9, pp. 97–114, 1994.
- [17] J. Korczak and N. Louis, "Synthesis of conceptual hierarchies applied to remote sensing," in *Image and Signal Processing for Remote Sensing IV*, vol. 3500 of *Proceedings of SPIE*, pp. 397–406, Barcelona, Spain, September 1999.
- [18] J. Korczak and A. Quirin, "Evolutionary approach to discovery of classification rules from remote sensing images," in *Proceedings of the 5th European Workshop on Evolutionary Computation in Image Analysis and Signal Processing (EvoIASP '03)*, pp. 338–398, Essex, UK, April 2003.
- [19] T. Kurita and N. Otsu, "Texture classification by higher order local autocorrelation features," in *Proceedings of the 1st Asian Conference on Computer Vision (ACCV '93)*, pp. 175–178, Osaka, Japan, November 1993.
- [20] P. L. Lanzi, "A study of the generalization capabilities of XCS," in *Proceedings of the 7th International Conference on Genetic Algorithms (ICGA '97)*, pp. 418–425, Morgan Kaufmann, East Lansing, Mich, USA, July 1997.
- [21] X. Liu, K. Chen, and D. Wang, "Extraction of hydrographic regions from remote sensing images using an oscillator network with weight adaptation," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 39, no. 1, pp. 207–211, 2001.
- [22] M. T. Manry, S. J. Apollo, L. S. Allen, et al., "Fast training of neural networks for remote sensing," *Remote Sensing Reviews*, vol. 9, no. 1-2, pp. 77–96, 1994.
- [23] E. Ozcan and C. K. Mohan, "Partial shape matching using genetic algorithms," *Pattern Recognition Letters*, vol. 18, no. 10, pp. 987–992, 1997.
- [24] A. Quirin, "Discovery of classification rules : evolutionary classifiers," M.S. thesis, Louis Pasteur University, LSIIT/CNRS, Strasbourg, France, 2002.
- [25] M. V. Rendon, "Reinforcement learning in the fuzzy classifier system," Reporte de Investigaci CIA-RI-031, ITESM, Campus Monterrey, Centro de Inteligencia Artificial, Monterrey, Mexico, 1997.
- [26] R. A. Richards, *Zeroth-Order Shape Optimization Utilizing a Learning Classifier System*, Stanford University, Stanford, Calif, USA, 1995, <http://www.stanford.edu/~buc/SPHINCsX/book.html>.
- [27] R. L. Riolo, "Empirical studies of default hierarchies and sequences of rules in learning classifier systems," Ph.D. dissertation, Computer Science and Engineering Department, University of Michigan, Dearborn, Mich, USA, 1988.
- [28] B. J. Ross, A. G. Gualtieri, F. Fueten, and P. Budkewitsch, "Hyperspectral image analysis using genetic programming," in *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO '02)*, pp. 1196–1203, New York, NY, USA, July 2002.
- [29] D. A. Stow, A. Hope, D. McGuire, et al., "Remote sensing of vegetation and land-cover change in arctic tundra ecosystems," *Remote Sensing of Environment*, vol. 89, no. 3, pp. 281–308, 2004.
- [30] T. Toutin and P. Cheng, "Quickbird—a milestone for highresolution mapping," *Earth Observation Magazine*, vol. 11, no. 4, pp. 14–18, 2002.

- [31] D. Verbyla and T. Hammond, "How to lie with an error matrix," *Remote Sensing & Image Analysis*, 2002, <http://nrm.salrm.uaf.edu/~dverbyla/online/errormatrix.html>.
- [32] C. Weber, *Images Satellitaires et Milieu Urbain*, Hermès, Paris, France, 1995.
- [33] S. W. Wilson, "Classifier fitness based on accuracy," *Evolutionary Computation*, vol. 3, no. 2, pp. 149–175, 1995.
- [34] S. W. Wilson, "Generalization in the XCS classifier system," in *Proceedings of the 3rd Annual Genetic Programming Conference*, pp. 665–674, Madison, Wis, USA, July 1998.
- [35] S. W. Wilson, "State of XCS classifier system research," in *Proceedings of the 2nd International Workshop on Learning Classifier Systems (IWLCS '99)*, pp. 63–81, Orlando, Fla, USA, July 1999.
- [36] S. W. Wilson, "Get real! XCS with continuous-valued inputs," in *Learning Classifier Systems: From Foundations to Applications*, vol. 1813 of *Lecture Notes in Computer Science*, pp. 209–220, Springer, London, UK, 2000.

Arnaud Quirin: European Centre for Soft Computing, Mieres 33600, Spain
Email: arnaud.quirin@softcomputing.es

Jerzy Korczak: Laboratoire des Sciences de l'Image, de l'Informatique et de la Télédétection, Centre National de la recherche scientifique, Université Louis Pasteur, Strasbourg, France
Email: jjk@dpt-info.u-strasbg.fr

17

Genetic programming techniques for multiclass object recognition

Mengjie Zhang

17.1. Introduction

Classification tasks arise in a very wide range of applications, such as detecting faces from video images, recognising words in streams of speech, diagnosing medical conditions from the output of medical tests, and detecting fraudulent credit card fraud transactions [11, 15, 50]. In many cases, people (possibly highly trained experts) are able to perform the classification task well, but there is either a shortage of such experts, or the cost of people is too high. Given the amount of data that needs to be classified, automatic computer-based classification programmes/systems are of immense social and economic value.

A classification programme must correctly map an input vector describing an instance (such as an object image) to one of a small set of class labels. Writing classification programmes that have sufficient accuracy and reliability are usually very difficult and often infeasible: human programmers often cannot identify all the subtle conditions needed to distinguish between all instances of different classes.

Genetic programming (GP) is a relatively recent and fast developing approach to automatic programming [4, 23, 25]. In GP, solutions to a problem can be represented in different forms but are usually interpreted as computer programmes. Darwinian principles of natural selection and recombination are used to evolve a population of programmes towards an effective solution to specific problems. The flexibility and expressiveness of computer programme representation, combined with the powerful capabilities of evolutionary search, make GP an exciting new method to solve a great variety of problems. A strength of this approach is that evolved programmes can be much more flexible than the highly constrained, parameterised models used in other techniques such as neural networks and support vector machines. GP has been applied to a range of object recognition tasks such as shape classification, face identification, and medical diagnosis with some success.

GP research has considered a variety of kinds of classifier programmes, using different programme representations, including tree or tree-like classifiers, decision tree classifiers, classification rule sets [25], and linear and graph classifiers

[4]. Recently, tree-like numeric expression representation classifiers have been developed using GP [30, 42, 46, 58]. In these years, this form has been successfully applied to some real-world classification problems such as detecting and recognising particular classes of objects in images [42, 46, 57, 59], demonstrating the potential of GP as a general method for classification problems.

Tree-like numeric expression GP classifiers model a solution to a classification problem in the form of a mathematical expression, using a set of arithmetic and mathematical operators, possibly combined with conditional/logic operators such as the “if-then-else” structures commonly used in computer programmes.

The output of a tree-like numeric expression GP classifier is a numeric value that is typically translated into a class label. For the simple binary classification case, this translation can be based on the sign of the numeric value [17, 30, 37, 38, 42, 46, 60]; for multiclass problems, finding the appropriate boundary values to separate the different classes is more difficult. The simplest approach—fixing the boundary values at manually chosen points—often results in unnecessarily complex programmes and could lead to poor performance and very long training times [30, 58, 59].

17.1.1. Goals

To avoid these problems and to improve the classification accuracy and the efficiency of the GP system, this chapter aims to develop better classification strategies in GP for multiclass object classification problems. The main focus is on the translation of the numeric output of a genetic programme classifier into class labels. Rather than using manually predefined boundary values, we will consider new methods which allow each genetic programme to use a set of dynamically determined class boundaries. We will compare the dynamic methods with the current static (manually defined) method on a sequence of image classification problems of increasing difficulty.

17.1.2. Organisation

This chapter is organised as follows. Section 17.2 describes the field of object recognition and some essential background of recent GP-related work for object recognition problems. Section 17.3 describes the overall GP approach for object classification problems. Section 17.4 describes the class translation rules. Section 17.5 presents the five image classification problems examined in this approach. Section 17.6 presents the experimental results and Section 17.7 gives the concluding remarks.

17.2. Background

This section briefly describes the field of the object recognition with traditional approaches, then presents recent GP-related work to object recognition problems.

17.2.1. Object recognition

Object recognition, also called automatic object recognition or automatic target recognition, is a specific field and a challenging problem in computer vision and image understanding [12]. This task often involves *object localisation* and *object classification*. Object localisation refers to the task of identifying the positions of the objects of interest in a sequence of images either within the visual or infrared spectral bands. Object classification refers to the task of discriminating between images of different kinds of objects, where each image contains only one of the objects of interest.

Traditionally, most research on object recognition involves four stages: *pre-processing*, *segmentation*, *feature extraction*, and *classification* [6, 13]. The pre-processing stage aims to remove noise or enhance edges. 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. 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 to object classification.

Object recognition has been of tremendous importance in many application domains. These domains include military applications [16, 46, 55], human face recognition [48, 50], agricultural product classification [54], handwritten character recognition [1, 26], medical image analysis [51], postal code recognition [9, 27], and texture classification [43].

Since the 1990s, many methods have been employed for object recognition. These include different kinds of neural networks [2, 9, 44, 52], genetic algorithms [3, 20], decision trees [35], statistical methods such as Gaussian models and naive Bayes [10, 35], support vector machines [10, 35], genetic programming [17, 19, 48, 53], and hybrid methods [8, 22, 56].

17.2.2. GP-related work to object recognition

Since the early 1990s, there has been only a relatively small amount of work on applying genetic programming techniques to object recognition, including object classification and object localisation. This in part reflects the fact that genetic programming is a relatively young discipline compared with, say, neural networks. This subsection briefly reviews the GP work related to object recognition.

Song [39–43] uses tree-based genetic programming for a series of object image texture classification problems, such as classification of bitmap patterns, Brodatz textures, and mashing images. This work mainly focuses on the use of GP for binary classification problems. There are also some multiclass problems in this work but he decomposes the multiclass problems into multiple binary classification problems then applies GP to these binary problems. Both features and pixel

values are used as terminals, the four standard arithmetic operators and a conditional operator with some relational operators are used to construct the function set, and the classification accuracy is used as the fitness function. The results show that, “with an appropriate methodology, GP can be used as a texture classification method without computationally expensive feature extraction.”

Loveard [29, 30] uses strongly typed genetic programming for a number of object recognition problems, including classification of medical images and satellite image objects. This work investigates a number of classification strategies. His results show that the dynamic range selection strategy outperformed other strategies on those classification problems.

Ciesielski et al. [7] use genetic programming for a real world object detection problem—finding orthodontic landmarks in crano-facial X-Rays. The system could evolve genetic programmes to implement a linear function of the features. Analysis of these linear functions reveals that underlying regularities can be captured. The analysis also suggests that evolved algorithms are a realistic solution to the object detection problem, given the features and operators available.

Tackett [46, 47] uses genetic programming to assign detected image features to a *target* or *nontarget* category. Seven primitive image features and twenty statistical features are extracted and used as the terminal set. The four standard arithmetic operators and a logic function are used as the function set. The fitness function is based on the classification result. The approach was tested on US Army NVEOD terrain board imagery, where vehicles such as tanks need to be classified. The genetic programming method outperformed both a neural network classifier and a binary tree classifier on the same data, producing lower rates of false positives for the same detection rates.

Andre [1] uses genetic programming to evolve functions that traverse an image, calling upon coevolved detectors in the form of hit-miss matrices to guide the search. These hit-miss matrices are evolved with a two-dimensional genetic algorithm. These evolved functions are used to discriminate between two letters or to recognise single digits.

Koza [25, Chapter 15] uses a “turtle” to walk over a bitmap landscape. This bitmap is to be classified either as a letter “L,” a letter “I,” or neither of them. The turtle has access to the values of the pixels in the bitmap by moving over them and calling a detector primitive. The turtle uses a decision tree process, in conjunction with negative primitives, to walk over the bitmap and decide which category a particular landscape falls into. Using automatically defined functions as local detectors and a constrained syntactic structure, some perfect scoring classification programmes were found. Further experiments showed that detectors can be made for different sizes and positions of letters, although each detector has to be specialised to a given combination of these factors.

Teller and Veloso [48] use a genetic programming method based on the PADO language to perform face recognition tasks on a database of face images in which the evolved programmes have a local-indexed memory. The approach was tested on a discrimination task between 5 classes of images [49] and achieved up to 60% correct classification for images without noise.

Robinson and McIlroy [34] apply genetic programming techniques to the problem of eye location in grey-level face images. The input data from the images is restricted to a 3000-pixel block around the location of the eyes in the face image. This approach produced promising results over a very small training set, up to 100% true positive detection with no false positives, on a three-image training set. Over larger sets, the genetic programming approach performed less well however, and could not match the performance of neural network techniques.

Winkeler and Manjunath [53] produce genetic programmes to locate faces in images. Face samples are cut out and scaled, then preprocessed for feature extraction. The statistics gleaned from these segments are used as terminals in genetic programming which evolves an expression returning how likely a pixel is to be part of a face image. Separate experiments process the grey scale image directly, using low-level image processing primitives and scale-space filters.

Zhang et al. [38, 57–60] uses genetic programming for a number of object classification and detection problems. Typically, low-level pixel statistics are used to form the terminal set, the four arithmetic operators are used to construct the function set, and the fitness functions are based on either classification accuracy or error rate for object classification problems, and detection rate and false alarm rate for object localisation and detection problems. Good results have been achieved on classification and detection of regular objects against a relatively uncluttered background.

Since the work to be presented in this paper focuses on the use of genetic programming techniques for object recognition, Table 17.1 lists the recent research to overview the GP-related work based on the applications and the first authors.

17.3. The GP approach for object classification

In this approach, we used the numeric-expression-based tree structure to represent genetic programmes. The ramped half-and-half method was used for generating the programmes in the initial population and for the mutation operator. The proportional selection mechanism and the reproduction, crossover, and mutation operators were used in the learning and evolutionary process.

In the remainder of this section, we address the other aspects of the GP learning/evolutionary system: (1) determination of the terminal set, (2) determination of the function set, and (3) construction of the fitness measure.

17.3.1. Terminals

For object classification problems, terminals generally correspond to image features. Some conventional approaches to image recognition usually use high-level, domain specific features of images as inputs to a learning/classification system, which generally involves a time consuming feature selection and a hand crafting

TABLE 17.1. Object recognition related work based on genetic programming.

Problems	Applications	Authors	Source
Object Classification	Tank detection	Tackett	[46, 47]
	Letter recognition	Andre Koza	[1] [25]
	Face recognition	Teller and Veloso	[48]
	Small target classification	Stanhope and Daida	[45]
	Shape recognition	Winkeler and Manjunath	[53]
	Eye recognition	Teller and Veloso	[49]
	Texture classification	Robinson and McIlroy	[34]
	Medical object classification	Song et al.	[39–43]
Object Detection	Medical object classification	Loveard and Ciesielski	[29, 30]
	Shape and coin recognition	Zhang and Smart	[60]
	Orthodontic landmark detection	Ciesielski et al.	[7]
	Ship detection	Howard et al.	[17]
	Mouth detection	Isaka	[21]
	Small target detection	Benson	[5]
Other Vision Problems	Vehicle detection	Howard et al.	[19]
	Medical object detection	Zhang et al.	[58, 59]
	Edge detection	Lucier et al.	[31]
	San Mateo trail problem	Koza Koza	[23] [24]
	Image analysis	Howard et al. Poli	[18] [33]
	Model Interpretation	Lindblad et al.	[28]
	Stereoscopic Vision	Graae et al.	[14]
	Image compression	Nordin and Banzhaf	[32]

of feature extraction programmes. In this approach, we used pixel level, domain-independent statistical features (referred to as pixel statistics) as terminals. We expect that the GP evolutionary process can automatically select features that are relevant to a particular domain to construct good genetic programmes.

Four pixel statistics are used in this approach: the average intensity of the whole object image, the variance of intensity of the whole object image, the average intensity of the central local region, and the variance of intensity of the central local region, as shown in Figure 17.1.

Since the range of these four features are quite different, we linearly normalised these feature values into the range $[-1, 1]$ based on all object image examples to be classified.

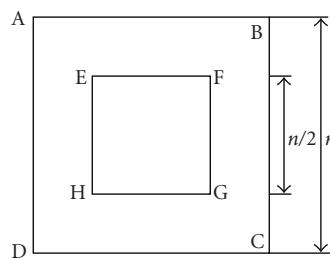


FIGURE 17.1. Region features as terminals.

Notice that these features might not be sufficient for some difficult object classification problems. However, they have been found reasonable in many problems and the selection of good features is not the goal of this chapter.

In addition, we also used some constants as terminals. These constants are randomly generated using a uniform distribution. To be consistent with the feature terminals, we also set the range of the constants as $[-1, 1]$. Unlike the feature terminals where the same feature usually has different values for different object images, the constant terminals will remain unchanged for all object images through the whole evolutionary process.

17.3.2. Functions

In the function set, the four-standard arithmetic and a conditional operation were used to form the function set

$$\text{FuncSet} = \{+, -, \times, /, \text{if} \}. \quad (17.1)$$

The $+$, $-$, and \times operators have their usual meanings: addition, subtraction, and multiplication, while $/$ represents “protected” division which is the usual division operator except that a divide by zero gives a result of zero. Each of these functions takes two arguments. The *if* function takes three arguments. The first argument, which can be any expression, constitutes the condition. If the first argument is negative, the *if* function returns its second argument; otherwise, it returns its third argument. The *if* function allows a programme to contain a different expression in different regions of the feature space, and allows discontinuous programmes, rather than insisting on smooth functions.

17.3.3. Fitness function

We used classification accuracy on the training set of object images as the fitness function. The classification accuracy of a genetic programme classifier refers to the number of object images that are correctly classified by the genetic programme classifier as a proportion of the total number of object images in the training set. According to this design, the best fitness is 100%, meaning that all object images

have been correctly recognised without any missing objects or any false alarms for any class.

To calculate the classification accuracy of a genetic programme, one needs to determine how to translate the single programme output to a set of class labels. This is described in Section 17.4. Image datasets, experiment parameters, and the termination strategy will be described in Section 17.5.

17.4. Translation rules in classification

As mentioned earlier, each evolved genetic programme has a numeric output value, which needs to be translated into class labels. The methods which perform this translation are referred to as *class translation rules* in this chapter.

This section briefly describes the static class boundary determination (SCBD) method [58] for multiclass classification commonly used in many approaches [29, 30, 39, 42], then details the two new class translation rules: centred dynamic class boundary determination (CDCBD) and slotted dynamic class boundary determination (SDCBD).

17.4.1. Static class boundary determination

Introduced in [58], the static class boundary determination (SCBD) method has been used in many approaches to classification problems with three or more classes. In this method, two or more predefined thresholds/boundaries are applied to the numeric output value of the genetic programme and the ranges/regions between these boundaries are linearly 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 $n - 1$, as shown in Figure 17.2.

In this figure, n refers to the number of object classes, v is the output value of the evolved programme, T_1, T_2, T_{n-1} are static, predefined class boundaries, and F_1, F_2 , and F_3 are input features for a particular object classification task (forming feature terminals).

Although this class translation rule is easy to set and has achieved some success in several problems [39, 59], it has a number of disadvantages. Firstly, the ordering of classes is fixed. For binary classification problems, we only need one boundary value (usually zero) to separate the programme 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. Secondly, the regions along the programme output space are also fixed for these

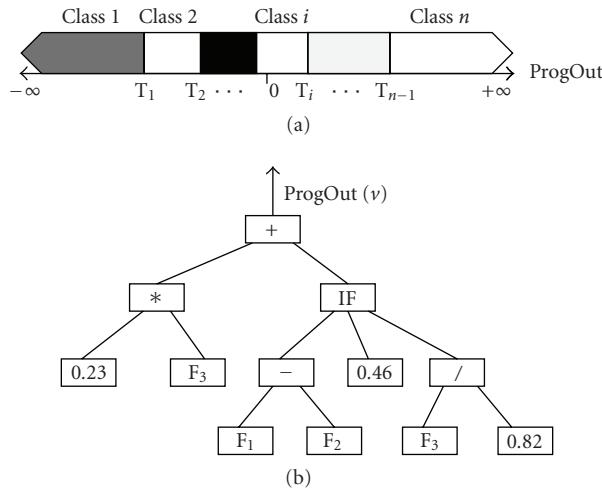


FIGURE 17.2. Static class boundary determination method.

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.

17.4.2. Centred dynamic class boundary determination

The first new method is the centred dynamic class boundary determination (CD-CBD), where the class boundaries are dynamically determined by calculating the centre of the programme output value for each class. An example of the process in this method is shown in Figure 17.3.

The algorithm is presented as follows.

Step 1. Initialise the class boundaries as certain predefined values as in the SCBD method.

Step 2. Evaluate each genetic programme in the population to obtain the programme output value for each training example based on the SCBD method.

During the evolutionary process, repeat step 3 and step 4.

Step 3. For each class c , calculate the centre of the class according to (17.2):

$$\text{Center}_c = \frac{\sum_{p=1}^M \sum_{\mu_c=1}^L (W_p \times \text{ProgOut}_{p\mu_c})}{\sum_{p=1}^M \sum_{\mu_c=1}^L W_p}, \quad (17.2)$$

where M is the number of programmes in the population and p is the index, L is the number of training examples for class c and μ_c is the index, $\text{ProgOut}_{p\mu_c}$ is the output value of the p th programme on training example μ_c for class c , and W_p

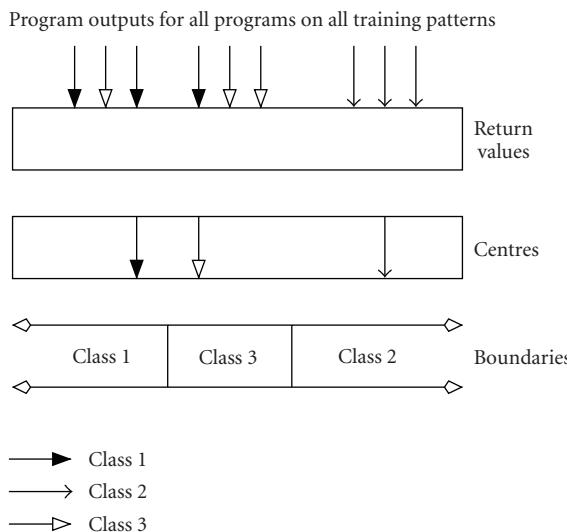


FIGURE 17.3. An example of the CDCBD method.

is a weighting factor which reflects the relative importance or contribution of the programme p over all the programmes in the population and is calculated by

$$W_p = \frac{2 \times (\text{fit}_p \times (K - 1) + 1)}{K + 1}, \quad (17.3)$$

where fit_p is the raw fitness (classification accuracy) of programme p on all the examples in the training set, and K is a parameter defined by the user. Clearly, different values of K will result in different relative contribution of the programme. In this work, we used a value 3 for K , reflecting that a good programme is considered more important than a bad programme.

Step 4. Calculate the boundary between every two classes by taking the middle point of the two adjacent class centres.

Step 5. Classify the training examples based on the class boundaries and calculate the new fitness (classification accuracy) of each genetic programme.

While this method could be applied to every generation of the evolutionary process, we applied this method to the training examples every five generations to keep balance between evolution and class boundary determination.

17.4.3. Slotted dynamic class boundary determination

The second new class translation rule is slotted dynamic class boundary determination (SDCBD). In this method, the output value of a programme is split into certain slots. When a large number of slots are used, a large amount of computation would be required. In our experiment, we used 100 slots derived from the range of $[-25, 25]$ with a step of 0.50. Since the input features (terminals) are

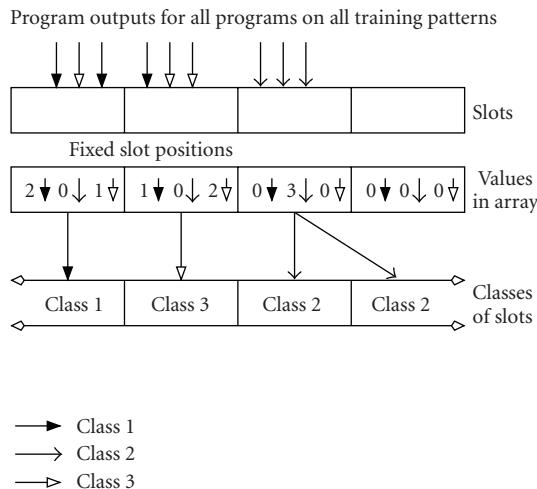


FIGURE 17.4. An example of the SDCBD method.

```

FOR each slot and each class
    Array[slot][class] = 0
FOR each training example X {
    FOR each programme p {
        ProgOut = execute programme p with X as input
        Round ProgOut to nearest slot
        IF ProgOut > 25 THEN ProgOut = 25
        IF ProgOut < -25 THEN ProgOut = -25
        Array[slot][class] += Wp
    }
}

```

ALGORITHM 17.1

scaled into $[-1, 1]$, the range $[-25, 25]$ is usually sufficient to represent the programme output. Each slot will be assigned to a value for each class. Figure 17.4 shows an example in the SDCBD process in a three-class problem.

In the first step, this method evaluates each genetic programme in the population to obtain the programme output value (ProgOut) for each training example and the fitness value of the programme based on the SCBD method.

In the second step, the method calculates the slot values for each class ($\text{Array[slot][class]}$) based on the programme output value. The algorithm for this step is as Algorithm 17.1, where W_p is calculated according to (17.3), reflecting the relative contribution of the genetic programme over all the programmes.

In the third step, this method dynamically determines to which class each slot belongs by simply taking the class with the largest value at the slot. However, in

```

FOR slot = 1 to 100 {
    FOR all class c {
        IF values of all class c in Array[slot][c] are zero {
            Class[slot] = ‘?’
        } ELSE {
            Search c for which Array[slot][c] is largest
            Class[slot] = c
        }
    }
    FOR slot = 1 to 100 {
        IF Class[slot] = ‘?’ {
            Class[slot] = nearest value to slot in the Class
            vector whose value is not ‘?’
        }
    }
}

```

ALGORITHM 17.2

case a slot does not hold any positive value, that is, no programmes produce any output at that slot for any training example, then this slot will be assigned to the class of the nearest neighbouring slot, as shown in Algorithm 17.2.

Similarly to the CDCBD method, this method is also applied to the evolutionary process every five generations so that at other generations the classification performance will be only improved based on the evolutionary process.

17.4.4. Characteristics of the dynamic methods

Compared with the SCBD method, these two new methods have the following characteristics.

- (i) The optimal boundaries for every two different classes or the optimal slot values for each class can be dynamically determined during the evolutionary process.
- (ii) Class labels do not have to fit into the predefined sequential regions. The optimal regions for each class in the programme output space can be automatically determined in the evolutionary process. For example, class 3 can be set in between class 1 and class 2 if necessary, as shown in Figures 17.3 and 17.4.
- (iii) When determining the new boundaries or slot values of classes for the next generation, the fitness values of the programme achieved in the current generation are explicitly taken into account. In this way, more heuristics are added to the evolutionary process, making the search method from the original “genetic beam” search to a kind of hybrid “genetic beam” and “hill-climbing” search.

With these new properties, we expect that these two methods would perform better on multiclass object classification problems, particularly for relatively difficult problems.

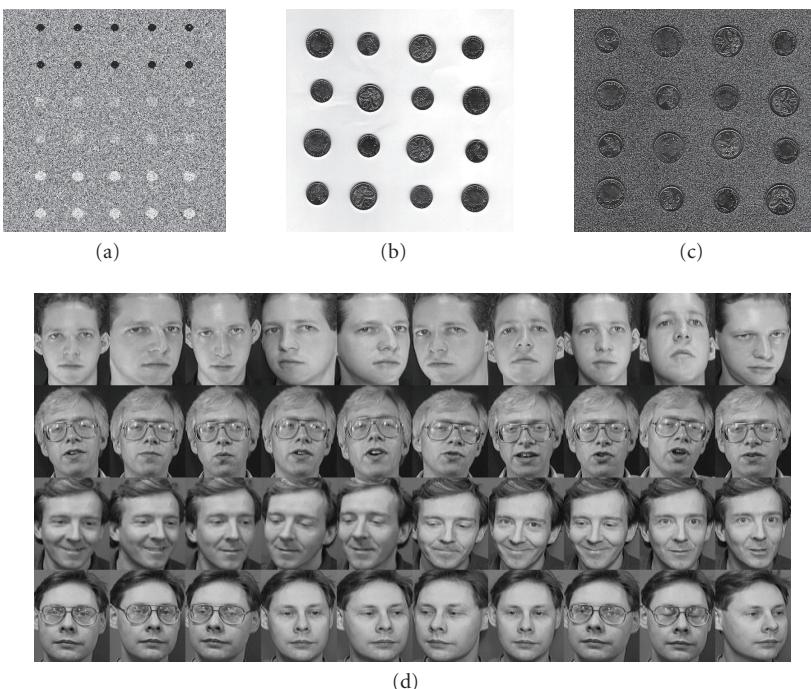


FIGURE 17.5. Example images from (a) Shape, (b) Coin1 and (c) Coin2, and (d) face.

17.5. Image datasets and experiment configurations

We used five image datasets in three groups with object classification problems of increasing difficulty in the experiments. Example images are shown in Figure 17.5. Experiment configurations will also be described in this section.

17.5.1. Computer-generated shape datasets

The first group of images (Figure 17.5(a)) was generated to give well-defined objects against a noisy background. The pixels of the objects were produced using a Gaussian generator with different means and variances for each class. Four classes of 712 small objects were cut out from those images to form the classification data. The four classes are black circles, light grey squares, white circles, and the grey noisy background.

Two different datasets, *shape1* and *shape2*, were constructed from this group of images. While set *shape1* arranges the four classes in an ordinary order based on the intensities, set *shape2* is intended to make these classes out of this order. Table 17.2 gives the class order and the initial setting of the boundaries between these classes. In the *Shape1* dataset, for example, classes 1, 2, 3, and 4 are arranged based on the ascending order of the intensities of the four classes. The class boundaries were set to -1.0 , 0 , and 1.0 , meaning that if a programme output value is less

TABLE 17.2. The class orders in the two shape datasets.

Dataset	Class label	Description	Intensity	Boundary
Shape1	1	Black circle	10 ± 5	-1.0 0 1.0
	2	Background	140 ± 50	
	3	Grey square	180 ± 50	
	4	White circle	220 ± 50	
Shape2	1	Background	140 ± 50	-1.0 0 1.0
	2	Black circle	10 ± 5	
	3	White square	220 ± 50	
	4	Grey square	180 ± 50	

than -1.0 for a particular object example, then this object example will be classified as class1 (black circles); if the programme output value is in $(-1.0, 0]$, it will be classified as class2 (background). Clearly, the classification problem is linearly separable in Shape1, but nonlinear separable in Shape2. The goal here is to investigate whether these classification methods perform well for the same data with the two different orders of class setting.

17.5.2. Coin datasets

The second group of images has two coin datasets. The first dataset (*coin1*, Figure 17.5(b)) consists of scanned 5-cent and 10-cent New Zealand coins. There are five classes of 576 objects: 5-cent heads, 5-cent tails, 10-cent heads and 10-cent tails, and a relatively uniform background. Compared with the *shape* dataset, the objects in this set (heads versus tails for either 5-cent or 10-cent coins) are more difficult to distinguish and there are more classes.

The second coin dataset (*coin2*, Figure 17.5(c)) also consists of five classes of objects, but the background is quite noisy, which makes the classification problems much harder. Even human eyes cannot perfectly distinguish these objects.

17.5.3. Face dataset

The fifth dataset consists of 40 human faces (Figure 17.5(d)) taken at different times, varying lighting slightly, with different expressions (open/closed eyes, smiling/nonsmiling) and facial details (glasses/no-glasses). These images were collected from the first four directories of the ORL face database [36]. All the images were taken against a dark homogeneous background with limited orientations. The task here is to distinguish those faces into the four different people.

17.5.4. Experiment configurations

For the shapes and the coins datasets, the objects were equally split into three separate datasets: one third for the training set used directly for learning the genetic

TABLE 17.3. Parameters used for GP training for the four datasets.

Parameter kinds	Parameter names	Shape1	Shape2	Coin1	Coin2	Faces
Search Parameters	Population-size	300	300	300	500	500
	Initial-max-depth	3	3	3	3	3
	Max-depth	5	5	6	8	8
	Max-generations	50	50	50	50	50
	Object-size	16×16	16×16	70×70	70×70	92×112
Genetic Parameters	Reproduction-rate	10%	10%	10%	10%	10%
	Cross-rate	60%	60%	60%	60%	60%
	Mutation-rate	30%	30%	30%	30%	30%
	Cross-term	15%	15%	15%	15%	15%
	Cross-func	85%	85%	85%	85%	85%

programme classifiers, one third for the validation set for controlling overfitting, and one third for the test set for measuring the performance of the learned programme classifiers. For the faces dataset, due to the small number of images, ten-fold cross validation was applied.

The parameter values used in this approach are shown in Table 17.3.

In this approach, the learning/evolutionary process is terminated when one of the following conditions is met.

- (i) The classification problem has been solved on the training set, that is, all objects of interest in the training set have been correctly classified without any missing objects or false alarms for any class.
- (ii) The accuracy on the validation set starts falling down.
- (iii) The number of generations reaches the predefined number, *max-generations*.

17.6. Results and discussion

This section presents a series of results of the two new dynamic class boundary determination methods on the five datasets in the shape, the coin, and the face image groups. These results are compared with those for the static class boundary method. For all experiments, we run 50 times and the average results (means and standard deviations) are presented.

17.6.1. Shape datasets

Table 17.4 shows the results of the three methods on the two shape datasets. The first line shows that for Shape1 dataset with 4 classes, the SCBD method achieved an average/mean accuracy of 99.67% over 50 runs on the test set and the average number of generations of the 50 runs spent on the training process was 12.32.

TABLE 17.4. Results on the shape datasets.

Dataset	Classes	Method	Gens	Accuracy (%)
Shape1	4	SCBD	12.32	99.67 \pm 0.79
		CDCBD	7.50	99.73 \pm 0.41
		SDCBD	8.43	99.68 \pm 0.45
Shape2	4	SCBD	50.0	96.87 \pm 1.24
		CDCBD	15.50	99.72 \pm 0.48
		SDCBD	24.30	99.35 \pm 0.54

TABLE 17.5. Results on the coin datasets.

Dataset	Classes	Method	Gens	Accuracy (%)
Coin1	5	SCBD	50.00	85.82 \pm 5.21
		CDCBD	37.70	96.10 \pm 2.34
		SDCBD	37.50	93.41 \pm 3.12
Coin2	5	SCBD	50.00	74.40 \pm 7.98
		CDCBD	39.32	91.60 \pm 3.79
		SDCBD	37.89	90.74 \pm 5.64
Face	4	SCBD	49.92	82.15 \pm 13.61
		CDCBD	46.86	92.45 \pm 12.22
		SDCBD	41.62	84.60 \pm 16.21

For the Shape1 dataset, all the three classification methods obtained nearly ideal results, reflecting the fact that this classification problem is relatively easy. In particular, the CDCBD method achieved the best performance.

For Shape2 data set, the two new dynamic methods gave very good results. However, the static SCBD method produced a much worse performance in both classification accuracy and training time¹ than the two new dynamic methods because the classes in this dataset were arranged arbitrary rather than in an ordinary order. This suggests that while the SCBD method could perform well on relatively easy, linearly separable classification problems with the classes arranged in an ordinary order, this method is not very appropriate for multiclass object classification problems with a randomly arranged order of classes. The two new dynamic methods, however, can be applied in this case. In addition, for these relatively easy classification problems, the CDCBD method seemed to be more effective and more efficient than the SDCBD method.

¹Note that the dynamic boundary determination process takes a bit longer time. However, since we apply the dynamic methods once every five generations and the computation cost of the two dynamic methods is quite low, the average time spent on each generation can be still considered similar.

17.6.2. Coin and face datasets

Table 17.5 shows the results of the three methods on the two coin datasets and the face dataset. These results show a similar pattern to the two shape datasets: the two dynamic methods achieved better results than the static method on all of these datasets in terms of both classification accuracy and training generations.

17.6.3. Summary and discussion

In summary, the results suggest that the SCBD method could perform well on relatively easy object classification problems if the classes were arranged in their ordinary order (such as Shape1), but would perform badly when the classes were out of this order (as in Shape2) or when the classification problems became more difficult (such as Coin1, Coin2, and Face datasets). This is mainly because a high degree of nonlinearity is required to map the class regions on the programme output to the object features in these situations.

The performances of all the three methods on Coin1, Coin2, and the Face datasets were worse than the two shape datasets, reflecting the fact that the classification problems in these datasets are more difficult than in the two shape datasets. Because these problems were harder, more features might need to be selected, extracted and added to the terminal set. Also more powerful functions might also need to be applied in order to obtain good performance. However, the investigation of these developments is beyond the goal and the scope of this chapter. We leave this for the future work.

In terms of the classification performance, the CDCBD method performed better than the SDCBD method for all of the datasets investigated here. In particular, the CDCBD method achieved over 90% of accuracy for the difficult problems in Coin2 and Face datasets, which was significantly better than the other two methods. Also notice that this method resulted in the least standard deviation over 50 runs among all the three methods investigated here. For training generations, it seems that the computational cost of the CDCBD method is also lower than the SDCBD method for the two shape datasets, but slightly higher for the three difficult problems in the coin and the face datasets. However, if a slightly higher computational cost can lead to clear improvement in classification performance, this will be a small price to pay in most situations.

17.7. Conclusions

The goal of this chapter was to investigate and explore dynamic class boundary determination methods as class translation rules in genetic programming for multiclass object classification problems, and to determine whether the new dynamic methods could outperform the static method. Two new classification methods, CDCBD and SDCBD, were developed and implemented where the class boundaries were dynamically determined during the evolutionary process.

The two dynamic methods were examined on five datasets in three object image groups providing object classification problems of varying difficulty. The

results showed that the static method, SCBD, performed very well on the relatively easy, linearly separable object classification problems where the classes were arranged in their ordinary order, but performed less well when the classes were arranged in an arbitrary order. The two dynamic methods, CDCBD and SDCBD, always outperformed the static method on all the object classification problems in the five datasets in terms of both classification performance and the training convergence. The centred dynamic method achieved better classification performance than the slotted dynamic method for all cases.

Although developed for object image classification problems, these two dynamic methods are also expected to be applied to other classification problems.

For future work, we will investigate whether the performance on the relatively difficult coin and face datasets can be improved if more features are added to the terminal set. We will also compare the performance with other long-term established methods such as decision trees and neural networks.

Acknowledgments

We would like to thank Peter Andreae for a number of useful discussions and Will Smart for some experiments. We would also like to thank the Artificial Intelligence Research Centre and College of Mechanical and Electrical Engineering, Agricultural University of Hebei for providing technical and system support. This work was supported in part by the Marsden Fund Council from Government funding, administered by the Royal Society of New Zealand under Grant no. 05-VUW-017 and the University Research Fund 06/9 at Victoria University of Wellington, New Zealand.

Bibliography

- [1] D. Andre, "Automatically defined features: the simultaneous evolution of 2-dimensional feature detectors and an algorithm for using them," in *Advances in Genetic Programming*, K. E. Kinnear, Ed., pp. 477–494, MIT Press, Cambridge, Mass, USA, 1994.
- [2] M. R. Azimi-Sadjadi, D. Yao, Q. Huang, and G. J. Dobeck, "Underwater target classification using wavelet packets and neural networks," *IEEE Transactions on Neural Networks*, vol. 11, no. 3, pp. 784–794, 2000.
- [3] J. Bala, K. De Jong, J. Huang, H. Vafaie, and H. Wechsler, "Using learning to facilitate the evolution of features for recognizing visual concepts," *Evolutionary Computation*, vol. 4, no. 3, pp. 297–311, 1996.
- [4] W. Banzhaf, P. Nordin, R. E. Keller, and F. D. Francone, *Genetic Programming: An Introduction on the Automatic Evolution of Computer Programs and Its Applications*, Dpunkt, Heidelberg, Germany, 1998.
- [5] K. Benson, "Evolving finite state machines with embedded genetic programming for automatic target detection," in *Proceedings of the IEEE Conference on Evolutionary Computation (CEC '00)*, vol. 2, pp. 1543–1549, IEEE Press, La Jolla, Calif, USA, July 2000.
- [6] T. Caelli and W. F. Bischof, *Machine Learning and Image Interpretation*, Plenum Press, New York, NY, USA, 1997.
- [7] V. Ciesielski, A. Innes, S. John, and J. Mamutil, "Understanding evolved genetic programs for a real world object detection problem," in *Proceedings of the 8th European Conference on Genetic Programming (EuroGP '05)*, M. Keijzer, A. Tettamanzi, P. Collet, J. I. van Hemert, and M.

- Tomassini, Eds., vol. 3447 of *Lecture Notes in Computer Science*, pp. 351–360, Springer, Lausanne, Switzerland, March-April 2005.
- [8] J. Riley and V. Ciesielski, “An evolutionary approach to training feed forward and recurrent neural networks,” in *Proceedings of the 2nd International Conference on Knowledge-Based Intelligent Electronic Systems (KES '98)*, L. C. Jain and R. K. Jain, Eds., vol. 3, pp. 596–602, Adelaide, Australia, April 1998.
 - [9] D. de Ridder, A. Hoekstra, and R. P. W. Duin, “Feature extraction in shared weights neural networks,” in *Proceedings of the 2nd Annual Conference of the Advanced School for Computing and Imaging (ASCI '96)*, pp. 289–294, Delft, The Netherlands, June 1996.
 - [10] M. H. Dunham, *Data Mining: Introductory and Advanced Topics*, Prentice-Hall, Englewood Cliffs, NJ, USA, 2003.
 - [11] J. Eggemont, A. E. Eiben, and J. I. van Hemert, “A comparison of genetic programming variants for data classification,” in *Proceedings of the 3rd Symposium on Intelligent Data Analysis (IDA '99)*, vol. 1642 of *Lecture Notes in Computer Science*, pp. 281–290, Springer, Amsterdam, The Netherlands, August 1999.
 - [12] D. A. Forsyth and J. Ponce, *Computer Vision: A Modern Approach*, Prentice-Hall, Englewood Cliffs, NJ, USA, 2003.
 - [13] E. Gose, R. Johnsonbaugh, and S. Jost, *Pattern Recognition and Image Analysis*, Prentice-Hall PTR, Upper Saddle River, NJ, USA, 1996.
 - [14] C. T. M. Graae, P. Nordin, and M. Nordahl, “Stereoscopic vision for a humanoid robot using genetic programming,” in *Real-World Applications of Evolutionary Computing*, S. Cagnoni, R. Poli, G. D. Smith, et al., Eds., vol. 1803 of *Lecture Notes in Computer Science*, pp. 12–21, Springer, Edinburgh, Tex, USA, April 2000.
 - [15] H. F. Gray, R. J. Maxwell, I. Martinez-Perez, C. Arus, and S. Cerdan, “Genetic programming for classification of brain tumours from nuclear magnetic resonance biopsy spectra,” in *Proceedings of the 1st Annual Conference on Genetic Programming (GP '96)*, J. R. Koza, D. E. Goldberg, D. B. Fogel, and R. L. Riolo, Eds., p. 424, MIT Press, Stanford University, Calif, USA, July 1996.
 - [16] A. Howard, C. Padgett, and C. C. Liebe, “A multi-stage neural network for automatic target detection,” in *Proceedings of IEEE International Joint Conference on Neural Networks. IEEE World Congress on Computational Intelligence (IJCNN '98)*, vol. 1, pp. 231–236, Anchorage, Alaska, USA, May 1998.
 - [17] D. Howard, S. C. Roberts, and R. Brankin, “Target detection in SAR imagery by genetic programming,” *Advances in Engineering Software*, vol. 30, no. 5, pp. 303–311, 1999.
 - [18] D. Howard, S. C. Roberts, and C. Ryan, “Evolution of an object detection ant for image analysis,” in *Genetic and Evolutionary Computation Conference Late Breaking Papers (GECCO '01)*, E. D. Goodman, Ed., pp. 168–175, San Francisco, Calif, USA, July 2001.
 - [19] D. Howard, S. C. Roberts, and C. Ryan, “The boru data crawler for object detection tasks in machine vision,” in *Proceedings of the Applications of Evolutionary Computing on EvoWorkshops: EvoCOP, EvoIASP, EvoSTIM/EvoPLAN*, S. Cagnoni, J. Gottlieb, E. Hart, M. Middendorf, and G. Raidl, Eds., vol. 2279 of *Lecture Notes in Computer Science*, pp. 222–232, Springer, Kinsale, Ireland, April 2002.
 - [20] J.-S. Huang and H.-C. Liu, “Object recognition using genetic algorithms with a Hopfield’s neural model,” *Expert Systems with Applications*, vol. 13, no. 3, pp. 191–199, 1997.
 - [21] S. Isaka, “An empirical study of facial image feature extraction by genetic programming,” in *Conference on the Genetic Programming*, J. R. Koza, Ed., pp. 93–99, Stanford University, Calif, USA, July 1997.
 - [22] P. G. Kornig, “Training neural networks by means of genetic algorithms working on very long chromosomes,” *International Journal of Neural Systems*, vol. 6, no. 3, pp. 299–316, 1995.
 - [23] J. R. Koza, *Genetic Programming: On the Programming of Computers by Means of Natural Selection*, MIT Press, Cambridge, Mass, USA, 1992.
 - [24] J. R. Koza, “Simultaneous discovery of reusable detectors and subroutines using genetic programming,” in *Proceedings of the 5th International Conference on Genetic Algorithms (ICGA '93)*, S. Forrest, Ed., pp. 295–302, Morgan Kauffman, Urbana-Champaign, Ill, USA, June 1993.

- [25] J. R. Koza, *Genetic Programming II: Automatic Discovery of Reusable Programs*, MIT Press, London, UK, 1994.
- [26] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," in *Intelligent Signal Processing*, pp. 306–351, IEEE Press, New York, NY, USA, 2001.
- [27] Y. LeCun, L. D. Jackel, B. Boser, et al., "Handwritten digit recognition: applications of neural network chips and automatic learning," *IEEE Communications Magazine*, vol. 27, no. 11, pp. 41–46, 1989.
- [28] R. Lindblad, P. Nordin, and K. Wolff, "Evolving 3d model interpretation of images using graphics hardware," in *Proceedings of the IEEE Conference on Evolutionary Computation (CEC '02)*, vol. 1, pp. 225–230, Honolulu, Hawaii, USA, May 2002.
- [29] T. Loveard, *Genetic programming for classification learning problems*, Ph.D. thesis, School of Computer Science and Information Technology, RMIT University, Melbourne, Australia, 2003.
- [30] T. Loveard and V. Ciesielski, "Representing classification problems in genetic programming," in *Proceedings of the IEEE Conference on Evolutionary Computation (CEC '01)*, vol. 2, pp. 1070–1077, IEEE Press, Seoul, Korea, May 2001.
- [31] B. J. Lucier, S. Mamillapalli, and J. Palsberg, "Program optimization for faster genetic programming," in *Proceedings of the 3rd Annual Conference on Genetic Programming (GP '98)*, pp. 202–207, Madison, Wis, USA, July 1998.
- [32] P. Nordin and W. Banzhaf, "Programmatic compression of images and sound," in *Proceedings of the 1st Annual Conference on Genetic Programming (GP '96)*, J. R. Koza, D. E. Goldberg, D. B. Fogel, and R. L. Riolo, Eds., pp. 345–350, MIT Press, Stanford University, Calif, USA, July 1996.
- [33] R. Poli, "Genetic programming for image analysis," in *Proceedings of the 1st Annual Conference on Genetic Programming (GP '96)*, J. R. Koza, D. E. Goldberg, D. B. Fogel, and R. L. Riolo, Eds., pp. 363–368, MIT Press, Stanford University, Calif, USA, July 1996.
- [34] G. Robinson and P. McIlroy, "Exploring some commercial applications of genetic programming," in *Evolutionary Computing*, T. C. Fogarty, Ed., vol. 993 of *Lecture Notes in Computer Science*, Springer, Berlin, Germany, 1995.
- [35] S. Russell and P. Norvig, *Artificial Intelligence, A Modern Approach*, Prentice-Hall, Englewood Cliffs, NJ, USA, 2nd edition, 2003.
- [36] F. S. Samaria and A. C. Harter, "Parameterisation of a stochastic model for human face identification," in *Proceedings of the 2nd IEEE Workshop on Applications of Computer Vision*, pp. 138–142, Sarasota, Fla, USA, December 1994.
- [37] J. R. Sherrah, R. E. Bogner, and A. Bouzerdoum, "The evolutionary pre-processor: automatic feature extraction for supervised classification using genetic programming," in *Proceedings of the 2nd Annual Conference on Genetic Programming (GP '97)*, J. R. Koza, K. Deb, M. Dorigo, et al., Eds., pp. 304–312, Morgan Kaufmann, Stanford University, Calif, USA, July 1997.
- [38] W. Smart and M. Zhang, "Classification strategies for image classification in genetic programming," in *Proceedings of Image and Vision Computing Conference*, D. Bailey, Ed., pp. 402–407, Palmerston North, New Zealand, November 2003.
- [39] A. Song, *Texture classification: a genetic programming approach*, Ph.D. thesis, Department of Computer Science, RMIT University, Melbourne, Australia, 2003.
- [40] A. Song and V. Ciesielski, "Fast texture segmentation using genetic programming," in *Proceedings of the IEEE Conference on Evolutionary Computation (CEC '03)*, R. Sarker, R. Reynolds, H. Abbass, et al., Eds., vol. 3, pp. 2126–2133, IEEE Press, Canberra, Australia, December 2003.
- [41] A. Song and V. Ciesielski, "Texture analysis by genetic programming," in *Proceedings of the IEEE Conference on Evolutionary Computation (CEC '04)*, vol. 2, pp. 2092–2099, IEEE Press, Portland, Ore, USA, June 2004.
- [42] A. Song, V. Ciesielski, and H. E. Williams, "Texture classifiers generated by genetic programming," in *Proceedings of the IEEE Conference on Evolutionary Computation (CEC '02)*, D. B. Fogel, M. A. El-Sharkawi, X. Yao, et al., Eds., vol. 1, pp. 243–248, IEEE Press, Honolulu, Hawaii, USA, May 2002.
- [43] A. Song, T. Loveard, and V. Ciesielski, "Towards genetic programming for texture classification," in *Proceedings of the 14th Australian Joint Conference on Artificial Intelligence (AI '01)*, pp. 461–472, Springer, Adelaide, Australia, December 2001.

- [44] C. Stahl and P. Schoppmann, "Advanced automatic target recognition for police helicopter missions," in *Automatic Target Recognition X*, F. A. Sadjadi, Ed., vol. 4050 of *Proceedings of SPIE*, pp. 61–68, Orlando, Fla, USA, April 2000.
- [45] S. A. Stanhope and J. M. Daida, "Genetic programming for automatic target classification and recognition in synthetic aperture radar imagery," in *Proceedings of the 7th Annual Conference on Evolutionary Programming*, V. W. Porto, N. Saravanan, D. Waagen, and A. E. Eiben, Eds., vol. 1447 of *Lecture Notes in Computer Science*, pp. 735–744, Springer, San Diego, Calif, USA, March 1998.
- [46] W. A. Tackett, "Genetic programming for feature discovery and image discrimination," in *Proceedings of the 5th International Conference on Genetic Algorithms (ICGA '93)*, S. Forrest, Ed., pp. 303–309, Morgan Kaufmann, Urbana-Champaign, Ill, USA, July 1993.
- [47] W. A. Tackett, *Recombination, selection, and the genetic construction of computer programs*, Ph.D. thesis, Faculty of the Graduate School, University of Southern California, Canoga Park, Calif, USA, April 1994.
- [48] A. Teller and M. Veloso, "A controlled experiment: evolution for learning difficult image classification," in *Proceedings of the 7th Portuguese Conference on Artificial Intelligence*, C. Pinto-Ferreira and N. J. Mamede, Eds., vol. 990 of *Lecture Notes in Artificial Intelligence*, pp. 165–176, Springer, Madeira Island, Portugal, October 1995.
- [49] A. Teller and M. Veloso, "PADO: learning tree structured algorithms for orchestration into an object recognition system," Tech. Rep. CMU-CS-95-101, Department of Computer Science, Carnegie Mellon University, Pittsburgh, Pa, USA, 1995.
- [50] D. Valentin, H. Abdi, and A. J. O'Toole, "Categorization and identification of human face images by neural networks: a review of linear auto-associator and principal component approaches," *Journal of Biological Systems*, vol. 2, no. 3, pp. 413–429, 1994.
- [51] B. Verma, "A neural network based technique to locate and classify microcalcifications in digital mammograms," in *Proceedings of IEEE International Joint Conference on Neural Networks. IEEE World Congress on Computational Intelligence (IJCNN '98)*, vol. 3, pp. 1790–1793, Anchorage, Alaska, USA, May 1998.
- [52] T. Wessels and C. W. Omlin, "A hybrid system for signature verification," in *Proceedings of the IEEE-INNS-ENNS International Joint Conference on Neural Networks (IJCNN '00)*, vol. 5, pp. 509–514, Como, Italy, July 2000.
- [53] J. F. Winkeler and B. S. Manjunath, "Genetic programming for object detection," in *Proceedings of the 2nd Annual Conference on Genetic Programming (GP '97)*, J. R. Koza, K. Deb, M. Dorigo, et al., Eds., pp. 330–335, Morgan Kaufmann, Stanford University, Calif, USA, July 1997.
- [54] P. Winter, W. Yang, S. Sokhansanj, and H. Wood, "Discrimination of hard-to-pop popcorn kernels by machine vision and neural network," in *ASAE/CSAE North-Central Intersectional Meeting*, pp. 96–107, Saskatoon, Canada, September 1996.
- [55] Y. C. Wong and M. K. Sundareshan, "Data fusion and tracking of complex target maneuvers with a simplex-trained neural network-based architecture," in *Proceedings of IEEE International Joint Conference on Neural Networks. IEEE World Congress on Computational Intelligence (IJCNN '98)*, vol. 2, pp. 1024–1029, Anchorage, Alaska, USA, May 1998.
- [56] X. Yao and Y. Liu, "A new evolutionary system for evolving artificial neural networks," *IEEE Transactions on Neural Networks*, vol. 8, no. 3, pp. 694–713, 1997.
- [57] M. Zhang, P. Andreae, and M. Pritchard, "Pixel statistics and false alarm area in genetic programming for object detection," in *Applications of Evolutionary Computing*, S. Cagnoni, Ed., vol. 2611 of *Lecture Notes in Computer Science*, pp. 455–466, Springer, Berlin, Germany, 2003.
- [58] M. Zhang and V. Ciesielski, "Genetic programming for multiple class object detection," in *Proceedings of the 12th Australian Joint Conference on Artificial Intelligence (AI '99)*, N. Foo, Ed., vol. 1747 of *Lecture Notes in Artificial Intelligence*, pp. 180–192, Springer, Sydney, Australia, December 1999.
- [59] M. Zhang, V. Ciesielski, and P. Andreae, "A domain-independent window approach to multi-class object detection using genetic programming," *EURASIP Journal on Applied Signal Processing*, vol. 2003, no. 8, pp. 841–859, 2003, special issue on genetic and evolutionary computation for signal processing and image analysis.

- [60] M. Zhang and W. Smart, "Multiclass object classification using genetic programming," in *Applications of Evolutionary Computing, EvoWorkshops: EvoBIO, EvoCOMNET, EvoHOT, EvoIASP, EvoMUSART, and EvoSTOC*, G. R. Raidl, S. Cagnoni, J. Branke, et al., Eds., vol. 3005 of *Lecture Notes in Computer Science*, pp. 369–378, Springer, Coimbra, Portugal, April 2004.

Mengjie Zhang: School of Mathematics, Statistics and Computer Science, Victoria University of Wellington, P.O. Box 600, Wellington 6140, New Zealand

Email: mengjie@mcs.vuw.ac.nz

18

Classification by evolved digital hardware

Jim Tørresen

18.1. Introduction

A number of automated procedures suited for design of image and signal classifiers have recently been developed. Some of these are based on evolvable hardware (EHW) and have been applied to a large range of real-world applications. The applications considered in this chapter are prosthetic hand control and traffic sign number recognition.

To enhance the lives of people who have lost a hand, prosthetic hands have existed for a long time. These are operated by the signals generated by contracting muscles—named electromyography (EMG) signals—in the remaining part of the arm [12]. Presently available systems normally provide only two motions: open and close hand grip. The systems are based on the user adapting *himself* to a fixed controller. That is, he must train himself to issue muscular motions triggering the wanted motion in the prosthetic hand. A long time is often required for rehabilitation.

By using EHW, it is possible to make the *controller* itself adapt to each disabled person. The controller is constructed as a pattern classification hardware which maps input patterns to the desired motions of the prosthetic hand. Adaptable controllers have been proposed based on neural networks [2]. These require a floating point processor or a neural network chip. EHW-based controllers, on the other hand, use a few layers of digital logic gates for the processing. Thus, a more compact implementation can be provided making it more feasible to be installed inside a prosthetic hand.

Experiments based on the EHW approach have already been undertaken by Kajitani et al. [7]. The research on adaptable controllers is based on designing a controller providing six different motions in three different degrees of freedom. Such a complex controller could probably only be designed by *adapting* the controller to each dedicated user. It consists of AND gates succeeded by OR gates (programmable logic array). The latter gates are the outputs of the controller, and the controller is evolved as one complete circuit. The simulation indicates a similar

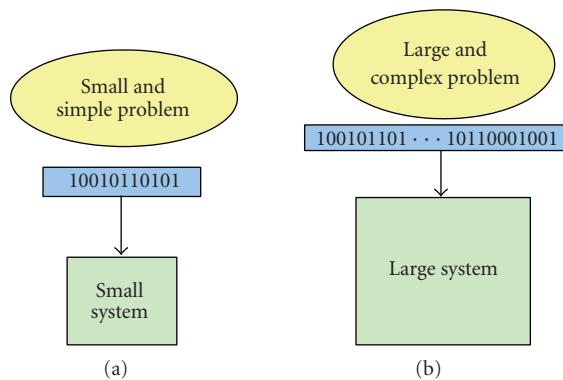


FIGURE 18.1. The chromosome string length and representation ability.

performance as artificial neural network but since the EHW controller requires a much smaller hardware, it is to be preferred. The approach proposed in this chapter is distinguished both with a more advanced architecture as well as applying incremental evolution.

An automatic traffic sign detection system would be important in a driver assistance system. In the second part of this chapter, an approach for detecting numbers on speed limit signs is proposed. Such a system would have to provide a high recognition performance in real time. The same architecture as applied for prosthetic hand control is used for classification of the numbers extracted from images. We are not aware of any other work combining evolvable hardware and road sign classification.

To make the system affordable, expensive hardware is not applicable. A promising technology which also allows for adaptation and upgrades is reconfigurable technology. Thus, we believe field programmable gate arrays (FPGA) would be an appropriate technology for hardware implementation. Other parts of the system would run as software on a processor.

One of the main problems in evolving hardware systems seems to be the limitation in the chromosome string length [10, 23]. A long string is normally required for solving a complex problem as seen in Figure 18.1. However, a larger number of generations is required by the evolutionary algorithm as the string increases. This often makes the search space becoming too large. Thus, work has been undertaken to try to diminish this limitation. Various experiments on speeding up the GA computation have been undertaken [1]. The schemes involve fitness computation in parallel or a partitioned population evolved in parallel—by parallel computation. Other approaches to the problem have been by using variable length chromosome [6] and reduced genotype representation [4]. Another option, called function-level evolution, is to evolve at a higher level than gate level [11]. Most work is based on fixed functions. However, there has been work in genetic programming for *evolving* automatically defined functions (ADF) [9].

Another improvement to artificial evolution, called coevolution, has been proposed [5]. In co-evolution, a part of the data, which defines the problem, co-evolves simultaneously with a population of individuals solving the problem. This could lead to a solution with a better generalization than a solution evolved based on the initial data. Further overview of related works can be found in [22].

Incremental evolution for EHW was first introduced in [16] for a character recognition system. The approach is a divide-and-conquer on the evolution of the EHW system, and thus, named *increased complexity evolution*. It consists of a division of the *problem* domain together with incremental evolution of the hardware system. This can be seen as dividing the problem on the righthand side in Figure 18.1 into a set of simpler and smaller problems—as given by the one on the lefthand side of the figure, which can be more easily evolved. Thus, evolution is first undertaken individually on a set of small systems, based on the assumption that the total evolution effort is less than for evolving a single system. The evolved small systems are the building blocks used in further evolution of a larger and more complex system. The benefits of applying this scheme is both a *smaller* and *smaller* search space compared to conducting evolution in one single run [22]. The goal is to develop a scheme that could evolve systems for complex real-world applications.

In this chapter, it is applied to evolve both a prosthetic hand controller circuit and for classifying numbers on speed limit signs. This is undertaken by a novel EHW architecture together with incremental evolution. The goal is to improve the generalization performance of gate-level EHW and make it a strong alternative to artificial neural networks.

The next two sections introduce the concepts of the evolvable hardware-based prosthetic hand controller. Then results are given in Section 18.4. The same EHW architecture and evolutionary scheme are then reused for speed limit sign number recognition in Section 18.5 followed by results in Section 18.6. Finally, conclusions are included in Section 18.7.

18.2. Prosthetic hand control

The research on adaptable controllers presented in this chapter provides control of six different motions in three different degrees of freedom: open and close hand, extension and flexion of wrist, and pronation and supination of wrist. The data set consists of the same motions as used in earlier work [7], and it has been collected by Dr. Kajitani at National Institute of Advanced Industrial Science and Technology (AIST) in Japan. The classification of the different motions could be undertaken by

- (a) *frequency domain*: the EMG input is converted by fast fourier transform (FFT) into a frequency spectrum,
- (b) *time domain*: the absolute value of the EMG signal is integrated for a certain time.

The latter scheme is used since the amount of computation and information are less than in the former.

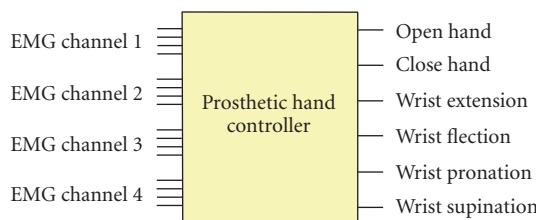


FIGURE 18.2. Illustration of the controller interfaces.

The published results on adaptive controllers are usually based on data for nondisabled persons. Since you may observe the hand motions, a good training set can be generated. For the disabled person, this is not possible since there is no hand observe. The person would have to distinguish the different motions by himself. Thus, it would be a harder task to get a high performance for such a training set but it will indicate the expected response to be obtainable by the prosthesis user. This kind of training set is applied in this chapter. Some of the initial results using this data set can be found in [18, 22].

18.2.1. Data set

The collection of the EMG signals are undertaken using three sensors for each channel. The difference in signal amplitude between the two of them, together with using the third as a reference, gave the resulting EMG signal. The absolute value of the EMG signal is integrated for 1 second and the resulting value is coded by *four* bits. To improve the performance of the controller, it is beneficial to be using several channels. In these experiments, *four* channels were used in total, giving an input vector of $4 \times 4 = 16$ bits. The controller interfaces are illustrated in Figure 18.2.

A subset of the training set input, consisting of preprocessed EMG signals, is given in Figure 18.3. For each motion, 10 samples are included.

The *output* vector consists of one binary output for each hand motion, and therefore, the output vector is coded by *six* bits. For each vector, only *one* bit is “1.” Thus, the data set is collected from a disabled person by considering *one* motion at a time in the following way:

- (1) the person contracts muscles corresponding to one of the six motions.
A personal computer (PC) samples the EMG signals;
- (2) the key corresponding to the motion is entered on the keyboard.

For each of the six possible motions, a total of 50 data vectors are collected, resulting in a total of: $6 \times 50 = 300$ vectors. Further, *two* such sets were made, one to be used for evolution (training) and the other to be used as a separate test set for validating the best circuit *after* evolution is finished.

Experiments have been undertaken to use a *lookup table* for solving this problem [21]. That is, each output vector in the data set is stored in a table addressed by the corresponding input vector. The data set is also used in various ways to

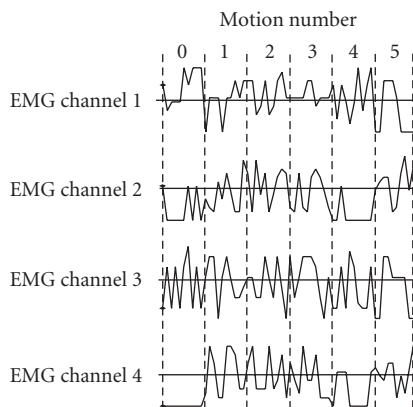


FIGURE 18.3. EMG signals from the training set. 10 samples of data for each motion.

program the rest of the empty locations in the table. In the best case, the test set performance was 58% (72.3% training set performance). The best training set performance obtained was 90%, but such a table gave a low test set performance.

18.3. An architecture for incremental evolution

In this section, the proposed architecture for the controller is described [18]. This includes the algorithms for undertaking the incremental evolution. This is all based on the principle of *increased complexity evolution* which was introduced in Section 18.1.

The architecture is illustrated in Figure 18.4. It consists of one subsystem for *each* of the six prosthetic motions. In each subsystem, the binary inputs x_0, \dots, x_{15} are processed by a number of different units, starting by the AND-OR unit. This is a layer of AND gates followed by a layer of OR gates. Each gate has the same number of inputs, and the number can be selected to be two, three, or four. The outputs of the OR gates are routed through a kind of filter called Selector. This unit selects *which* of these outputs that are to be counted by the succeeding Counter. That is, for each new input, the Counter is counting the number of *selected* outputs being "1" from the corresponding AND-OR unit. The main motivation for introducing the Selectors is to be able to select a set of outputs from each AND-OR unit in a flexible way to possibly improve the performance. Finally, the Max Detector outputs which counter, corresponding to *one* specific motion, is having the largest value. Each output from the Max Detector is connected to the corresponding motor in the prosthesis. If the Counter having the *largest* value corresponds to the correct hand motion, the input has been correctly classified.

A scheme, based on using multi-input AND gates together with Counters, has been proposed earlier [24]. However, the architecture used in this chapter is distinguished by including OR-gates, together with the Selector units involving incremental evolution.

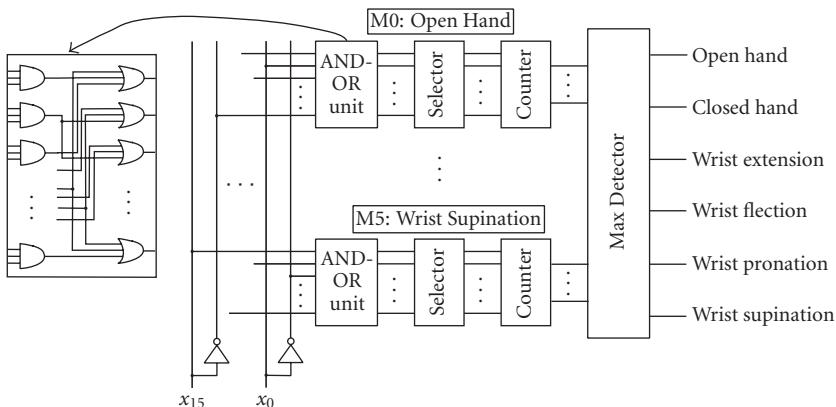


FIGURE 18.4. The digital gate-based architecture of the prosthetic hand controller.

The incremental evolution of this system can be described by the following steps.

- (1) *Step 1 evolution.* Evolve the AND-OR unit for each subsystem *separately*, one at a time. Apply *all* vectors in the training set for the evolution of each subsystem. There are no interactions among the subsystems at this step, and the fitness is measured on the outputs of the AND-OR units. A largest possible number of OR gates should be “1” for the 50 patterns corresponding to the motion the subsystem is set to respond to. For all other patterns, the number of gates outputting “1” should be as small as possible. That is, each subsystem should ideally respond only to the patterns for one specific prosthesis motion.
- (2) *Step 2 evolution.* Assemble the six AND-OR units into one system as seen in Figure 18.4. The AND-OR units are now fixed and the *Selectors* are to be evolved in the assembled system—in one common run. The fitness is measured using the same training set as in step 1 but the evaluation is now on the output of the Max Detector.
- (3) The system is now ready to be applied in the prosthesis.

In the first step, subsystems are evolved separately, while in the second step, these are assembled and evolved together by using the Selectors. The motivation for evolving separate subsystems, instead of a single system in one operation, is that earlier work has shown that the evolution time can be substantially reduced by this approach [16, 17].

The layers of AND and OR gates in one AND-OR unit consist of 32 gates each. This number has been selected to give a chromosome string of about 1000 bits which has been shown earlier to be appropriate. A larger number would have been beneficial for expressing more complex Boolean functions. However, the search space for evolution could easily become too large. For the step 1 evolution, each gate’s *input* is determined by evolution. The encoding of each gate in the binary

chromosome string is as follows:

Inp.1 (5 bit)	Inp.2 (5 bit)	(Inp.3 (5 bit))	(Inp.4 (5 bit))
---------------	---------------	-----------------	-----------------

As described in the previous section, the EMG signal input consists of 16 bits. Inverted versions of these are made available on the inputs as well, making up a total of 32 input lines to the gate array. The evolution is based on gate-level building blocks. However, since several output bits are used to represent one motion, the signal resolution becomes increased from the two binary levels.

For the step 2 evolution, each line in each Selector is represented by *one* bit in the chromosome string. This makes a chromosome string of 32×6 bits = 192 bits. If a bit is “0,” the corresponding line should *not* be input to the Counter, whereas if the bit is “1,” the line *should* be input.

18.3.1. Fitness measure

In step 1 evolution, the fitness is measured on all the 32 outputs of each AND-OR unit. In an alternative experiment, it was found that the performance could be substantially improved if the fitness is measured on a *limited* number (16 is here used as an example) of the outputs [18]. That is, each AND-OR unit still has 32 outputs but, as seen in Figure 18.5, only 16 are included in the computation of the fitness function:

$$\text{fitness} = \sum_{i=1}^{16} \text{Output OR gate } i \text{ match target } i. \quad (18.1)$$

The 16 outputs not used are included in the chromosome string and have *random* values. That is, their values do not affect the fitness of the circuit. After evolution, all the 32 outputs are applied for computing the performance:

$$\text{performance} = \sum_{i=1}^{32} \text{Output OR gate } i \text{ match target } i. \quad (18.2)$$

Since 16 OR gates are used for fitness computation, the “fitness measure” equals 16. In the figure, gates 1 to 16 are used for the fitness function. However, in principle, any 16 gates out of the 32 can be used. Other numbers than 16 were tested in experiments but 16 showed to give the best performance results and was used in the following reported experiments.

There are several possible benefits of this approach. Firstly, it could make the evolution easier find good circuits since there are less number of outputs to measure fitness on. Thus, it could lead to a higher training set performance. Secondly, this could be an interesting approach to improve the generalization of the circuit, that is, the test set performance. Only the OR gates in the AND-OR unit are “floating” during the evolution since all AND gates may be inputs to the 16 OR gates used by the fitness function. The 16 “floating” OR-gates (i.e., OR gates 17 to 32 in Figure 18.5) then provide additional combination of these *trained* AND gates.

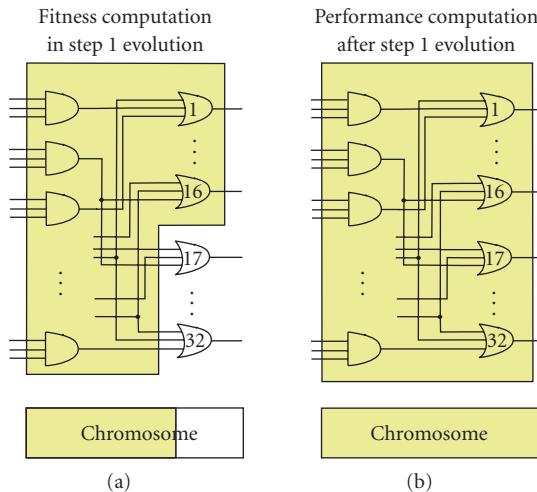


FIGURE 18.5. A “fitness measure” equal to 16.

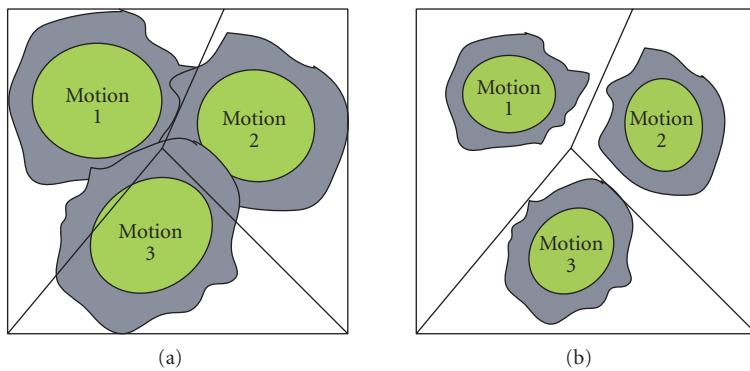


FIGURE 18.6. Illustration of noise added to (a) a plain signal and (b) a preprocessed signal.

Another way to look at this is that the “floating” gates provide “noise,” since the inputs to the “floating gates” are *randomly* connected to the AND gates. However, the noise is not added to the plain input but to a preprocessed and *improved* signal (output from the AND gates) as illustrated in Figure 18.6. The inner circle for each motion indicates the *training set* domain, with the outer circle indicating the added generalization obtained by adding “noise.” In (a), the signal is not pre-processed and adding noise makes the interference among classes worse while in (b), it improves the generalization rather than introducing interference. The step 2 evolution will be evolving the ratio of noise in the final system by adjusting the number of Selector bits set for gates 1 to 16 compared to the number set for gates 17 to 32.

The “floating” OR gates may also provide a *neutral fitness landscape* [8]. In such landscapes, the neutrality, movement between solutions with equal fitness [14], helps escaping local optimal solutions. The “floating” OR gates introduce dynamics into the population making this possible.

18.3.2. Fitness function

The fitness function is important for the performance when evolving circuits. For the step 1 evolution, the fitness function, applied for each AND-OR unit separately, is as follows for the motion m ($m \in [0, 5]$) unit:

$$F_1(m) = \frac{1}{s} \sum_{j=0}^{50m-1} \sum_{i=1}^O x + \sum_{j=50m}^{50m+49} \sum_{i=1}^O x + \frac{1}{s} \sum_{j=50m+50}^{P-1} \sum_{i=1}^O x, \quad (18.3)$$

where $x = \begin{cases} 0 & \text{if } y_{i,j} \neq d_{m,j}, \\ 1 & \text{if } y_{i,j} = d_{m,j}, \end{cases}$

where $y_{i,j}$ is the computed output of OR gate i , and $d_{m,j}$ is the corresponding target value of the training vector j . P is the total number of vectors in the training set ($P = 300$). As mentioned earlier, each subsystem is trained for one motion (the middle expression of F_1). This includes outputting “0” for input vectors for other motions (the first and last expressions of F_1).

The s is a scaling factor to implicitly emphasize on the vectors for the motion the given subsystem is assigned to detect. An appropriate value ($s = 4$) was found after some initial experiments. The O is the number of outputs included in the fitness function and is either 16 or 32 in the following experiments (referred to as “fitness measure” in the previous section).

The fitness function for the step 2 evolution is applied on the complete system and is given as follows:

$$F_2 = \sum_{j=0}^{P-1} x, \quad \text{where } x = \begin{cases} 1 & \text{if } d_{m,j} = 1, \text{ } m = i \text{ for which } \max_{i=0}^5 (\text{Counter}_i), \\ 0 & \text{else.} \end{cases} \quad (18.4)$$

This fitness function counts the number of training vectors for which the target *output* being “1” *equals* the *id* of the counter having the maximum output (as mentioned earlier only *one* output bit is “1” for each training vector).

18.3.3. The evolutionary algorithm

The simple genetic algorithm (GA), given by Goldberg [3], was applied for the evolution with a population size of 50. For each new generation, an entirely new population of individuals is generated. Elitism is used, thus the best individual from each generation is carried over to the next generation. The (single point) crossover rate is 0.8, thus the cloning rate is 0.2. Roulette wheel selection scheme

is applied. The mutation rate, the probability of bit inversion for each bit in the binary chromosome string, is 0.01. For some of the following experiments, other parameters have been used, but these are then mentioned in the text.

Various experiments were undertaken to find appropriate GA parameters. The ones that seemed to give the best results were selected and fixed for all the experiments. This was necessary due to the large number of experiments that would have been required if GA parameters should be able to vary through all the experiments. The preliminary experiments indicated that the parameter setting was not a major critical issue.

The proposed architecture fits into most FPGAs. The evolution is undertaken offline using software simulation. However, since no feedback connections are used and the number of gates between the input and output is limited, the real performance should equal the simulation. Any spikes could be removed using registers in the circuit. There are many benefits of being able to implement a system in hardware like cost, power consumption, and speed of operation.

For each experiment presented, four different runs of GA were performed. Thus, *each* of the four resulting circuits from step 1 evolution is taken to step 2 evolution and evolved for four runs.

18.4. Results

This section reports the experiments undertaken to search for an optimal configuration of the prosthetic hand controller. They will be targeted at obtaining the best possible performance for the *test* set.

18.4.1. Fitness measure approach

Table 18.1 shows the main results—in percentage of correct classification. Several different ways of evolving the controller are included. The training set and test set performances are listed on separate lines in the table. The “# inp/gate” column includes the number of inputs for each gate in the AND-OR units. The columns beneath “step 1 evolution” report the performance after only the *first* step of evolution. That is, each subsystem is evolved separately, and afterwards they become assembled to compute their total performance. The “Step 1+2 evolution” columns show the performance when the *Selector units* have been evolved as well (step 2 of evolution). In average, there is an improvement in the performance for the latter. Thus, the proposed *increased complexity evolution* give rise to improved performances.

In total, the best way of evolving the controller is the one listed first in the table. The circuit evolved with the best *test set* performance obtained, 67% correct classification. Figure 18.7 shows the step 2 evolution of this circuit. The training set performance is monotone increasing which is demanded by the fitness function. The test set performance is increasing with a couple of valleys. The circuit had a 60.7% test set performance after step 1 evolution.¹ Thus, the step 2 evolution

¹Evaluated with all 32 outputs of the subsystems.

TABLE 18.1. The results of evolving the prosthetic hand controller in several different ways.

Type of system	# inp/gate	Step 1 evolution			Step 1 + 2 evolution		
		Min	Max	Avr	Min	Max	Avr
A: Fitness measure 16 (train)	3	63.7	69.7	65.5	71.33	76.33	73.1
A: Fitness measure 16 (test)	3	50.3	60.7	55.7	44	67	55.1
B: Fitness measure 32 (train)	3	51	57.7	53.4	70	76	72.9
B: Fitness measure 32 (test)	3	40	46.7	44.4	45	54.3	50.1
C: Fitness measure 16 (train)	2	51.3	60.7	54.8	64.3	71.3	67.5
C: Fitness measure 16 (test)	2	46	51.7	49	44.3	54.7	50
D: Fitness measure 16 (train)	4	59.3	71.3	65.5	70	76	73.4
D: Fitness measure 16 (test)	4	52.7	59.7	55.3	48.3	56.3	52.7
E: Direct evolution (train)	4	56.7	63.3	59.3	—	—	—
E: Direct evolution (test)	4	32.7	43.7	36.6	—	—	—

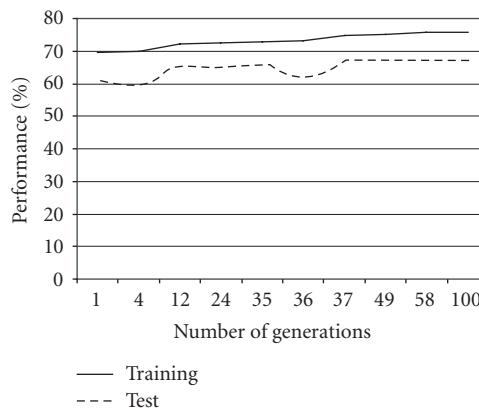


FIGURE 18.7. Plot of the step 2 evolution of the best performing circuit.

provides a substantial increase up to 67%. Other circuits did not perform that well, but the important issue is that it has been shown that the proposed architecture provides the *potential* for achieving high degree of generalization.

A feedforward neural network was trained and tested with the same data sets. The network consisted of (two weight layers with) 16 inputs, a variable number of hidden units, and 6 outputs. Table 18.2 shows the main results—in percentage of correct classification. Training was run for 1000 iterations, which were enough to find the maximum obtainable training set performance. The corresponding test set performance, measured after training is finished, is given in the column “Final.” However, to have a measure of the maximum possible test set performance, the test set performance was *monitored* throughout training and the maximum value is given in the column called “Maximum.” There is some difference between these two columns but it is not very large—especially for the networks with a large

TABLE 18.2. The results of neural network training of the prosthetic hand controller.

Number of hidden units	Training performance	Test set performance	
		Final	Maximum
20	88	54	60
30	86.7	54.3	59
40	88	58.7	60
50	86.7	57	60.3

number of hidden units. Thus, we see that it is beneficial to use a network with a large number of hidden units to obtain the highest training set performance.

In the best case, a test set performance of 60.3% correct classification was obtained. The best training set performance was 88%. Thus, a higher training set performance but a lower test set performance than for the best EHW circuit. The performance of the lookup table approach of 58%, reported in Section 18.2.1, is again less than that obtained by neural networks. This shows that the EHW architecture holds good generalization properties.

The experiment B is the same as A except that in B all 32 outputs of each AND-OR unit are used to compute the fitness function in the step 1 evolution. In A, each AND-OR unit also has 32 outputs but only 16 are included in the computation of the fitness function as described in Section 18.3.1. The performance of A in the table for the step 1 evolution is computed by using *all* the 32 outputs. Thus, over 10% better training set as well as the test set performance (in average) is obtained by having 16 outputs “floating” rather than measuring their fitness during the evolution!

What is also interesting is that if the performance of the circuits in A is measured (after step 1 evolution) with only 16 outputs, the performance is not very impressive. Thus, “floating” outputs in the evolution substantially improve the performance—including the *test* set performance. This may seem strange but has a reasonable explanation—given in Section 18.3.1. The reason for B having lower performance could be explained by a more complex as well as larger search space for the evolution. In this experiment, a larger number of OR gates should give a correct output and the chromosome string is longer since in A the bits assigned to the 16 “floating” OR-gates are not used. Other numbers of “floating” OR gates (8 and 24) were tested but the results were best for 16.

The C and D rows in the table contain the results when the gates in the AND-OR units each consists of two and four inputs, respectively. The lowest figures are for two input gates indicating that the architecture is too small to represent the given problem. Four inputs, on the other hand, could be too complex since having *three* input gates gives a slightly better results.

Each subsystem is evolved for 10 000 generations each, whereas the step 2 evolution was applied for 100 generation. These numbers were selected after a number of experiments. One comparison of the step 1 evolution (each gate having three inputs) is included in Figure 18.8 and shows that the best average performance is achieved when evolving for 10 000 generations.

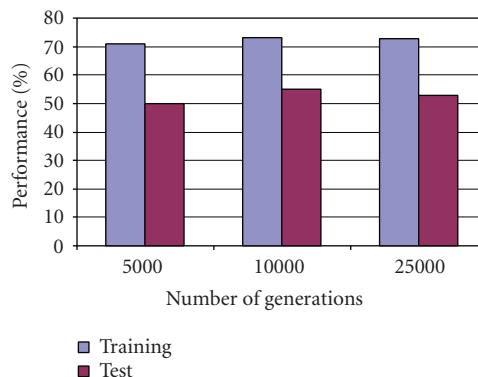


FIGURE 18.8. Performance of three different numbers of generations in step 1 evolution (average of four runs).

The circuits evolved with direct evolution (E) were evolved for 100,000 generations.² The training set performance is impressive when thinking of the simple circuit used. Each motion is controlled by a *single* four input OR gate. However, the test set performance is very much lower than what is achieved by the other approaches. This is explained by having an architecture that is too small to provide good generalization. A larger one, on the other hand, would make the chromosome string longer with the problems introduced in earlier (larger and more complex search space). This once again emphasizes the importance of applying the *increased complexity evolution* scheme.

18.5. Speed limit sign recognition

For the last years, there has been an increasing focus on enhancing traffic safety by applying intelligent systems in vehicles to assist drivers. One of the reasons for this becoming possible is the recent advance in computer hardware providing affordable technology with a high-processing capability. In the work presented in this chapter, we consider recognizing speed limit signs. Such a system could assist drivers on signs they did not notice before passing them. It will inform drivers about the present speed limit as well as possibly giving an alert if a car is driven faster than the speed limit. More actively, it could be applied to avoid using today's physical obstacles in the road. This would require that the system could control the vehicle so that it becomes impossible to drive faster than the speed limit. This will mainly be relevant on roads with low-speed limits. In the future, autonomous vehicles would have to be controlled by automatic road sign recognition. We have found very little work on speed limit sign classification. There exists a system based on global position system (GPS) and digital road maps with speed limits included [15]. However, such a system depends on much external infrastructure. Further,

²This is more than six times 10 000 which were used in the other experiments.



FIGURE 18.9. Speed limit signs.



FIGURE 18.10. An example of an input image.

problems like lack of updated speed limits on road maps question the reliability of such a system.

The work presented below concerns detection of the *numbers* on speed limit signs specifically. This is by classifying numbers extracted from the sign. This will be undertaken in digital logic gates configured by evolution as for the prosthetic hand controller.

18.5.1. Speed limit sign recognition system

Speed limit signs have features making them feasible for automatic detection.

First, there is a limited number of signs to distinguish. The following speed limit signs, see Figure 18.9, are used in the experiments: 30, 40, 50, 60, 70, and 80 km/h (other speed limit signs are not included due to lack of images). The outer circle of a sign is in *red* color.

Second, there are rules (named a road grammar) for how signs can be placed along the road. After an “80” sign, you will never find a “30” sign, but rather another “80” sign or a “60” sign. Third, the numbers on the signs are positioned vertically making the matching simpler. Only in curves the signs may be marginally tilted. Fourth, each time a *new* speed limit is given, there are speed limit signs at *both* sides of the driving lane. These features make it promising to undertake speed limit sign detection with a very high rate of correct prediction. A typical input image is shown in Figure 18.10.

The algorithm is divided into *three* parts: (1) image filtering to emphasize the red parts of the sign(s), (2) template matching to locate possible sign(s) in an image, and (3) sign number recognition [13, 19]. These will be presented below.

In the first part, a specialized robust color filter is applied on the image to emphasize the *red circle* surrounding the speed limit numbers. Further, this part of the algorithm effectively limits the number of red pixels in the image to be further processed.



FIGURE 18.11. Examples of extracted arrays from real images to be classified by EHW.

In the second part, possible signs are located in the image by searching for the *red circle* surrounding the numbers. The search is based on matching with a set of circular templates of various size. In addition to locating the sign, the algorithm tries to reject objects that are not signs and signs that are not speed limit signs. That is, to improve recognition at the same time as reducing the computation time.

18.5.2. Sign number recognition

The last part of the algorithm is to detect the speed limit *number* on a sign. This is conducted as follows.

- (1) Clean the space defined by the best template (remove RED and the surrounding colors), but keep the numbers.
- (2) Find boundaries of numbers (width and height).
- (3) Work only with the first number (the second is always zero).
- (4) Create a 7 (rows) \times 5 (columns) bit array of the given number (down-scaling). Set each bit of the array to 1 if there are more BLACK than WHITE pixels, else set the bit to 0.
- (5) Classify the bit array using the evolved classifier circuit (described in the next section).

Some randomly picked examples of bit arrays with different numbers are included in Figure 18.11.

18.5.2.1. Number of classification in EHW

To be able to correctly classify the bit array containing the extracted number, some kind of classification tool is needed. We would like to show that evolved digital logic gates are appropriate. The initial reason for this is the format of the array: it is *small* and contains *binary* values. If we applied other approaches like artificial neural network (ANN), a large number of floating point operations would be needed. The EHW architecture, on the other hand, could be implemented in combinational logic where classification is performed in parallel [20]. The next section presents the architecture applied for classification.

18.5.3. An evolvable architecture for classification

The classifier architecture is illustrated in Figure 18.12. It consists of one subsystem for *each* of the six numbers to be classified. In each subsystem, the binary inputs x_0, \dots, x_{31} (3 bits from the array are not used) are processed by a number of different units which are the same as for prosthetic hand controller. Further, the same incremental evolutionary algorithm is used.

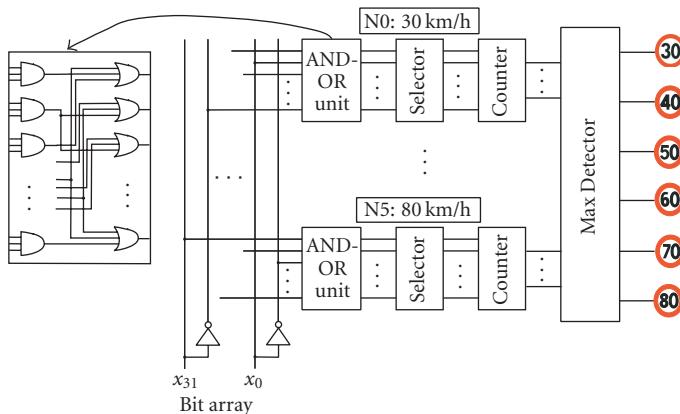


FIGURE 18.12. The digital gate-based architecture of the sign number classifier.

18.5.3.1. Fitness function

The fitness function is constructed in the same way as for the prosthetic hand controller including a “fitness measure” equal to 16. For the step 1 evolution, the fitness function, applied for each AND-OR unit separately, is as follows for the number n ($n \in \{0, \dots, 5\}$) unit:

$$F_1(n) = \frac{1}{s} \sum_{(n \text{ not active})} \sum_{i=1}^O x_i + \sum_{(n \text{ active})} \sum_{i=1}^O x_i, \quad \text{where } x_i = \begin{cases} 0 & \text{if } y_{i,j} \neq d_{n,j}, \\ 1 & \text{if } y_{i,j} = d_{n,j}, \end{cases} \quad (18.5)$$

where $y_{i,j}$ is the computed output of OR gate i , and $d_{n,j}$ is the corresponding target value of the training vector j . Each subsystem is trained for *one* number (the last expression of F_1). This includes outputting “0” for input vectors for other numbers (the first expressions of F_1). The s is a scaling factor to implicitly emphasize on the vectors for the number the given subsystem is assigned to detect. Since there is a variable number of training vectors for each number, this constant was set specifically for each subsystem (as seen in the result section). The O is the number of outputs included in the fitness function and is 16 in the following experiments (referred to as “fitness measure” in Section 18.3.1).

The fitness function for the step 2 evolution is applied on the complete system and is given as follows:

$$F_2 = \sum_{j=0}^{P-1} x_j, \quad \text{where } x_j = \begin{cases} 1 & \text{if } d_{n,j} = 1, \text{ } n = i \text{ for which } \max_{i=0}^5 (\text{Counter}_i), \\ 0 & \text{else.} \end{cases} \quad (18.6)$$

This fitness function counts the number of training vectors for which the target *output* being “1” *equals* the *id* of the Counter having the maximum output (only *one* output bit is “1” for each training vector). P is the total number of vectors in the training set ($P = 100$ in the following experiments).

18.5.3.2. Effective processing in EHW

In the last part of Section 18.5.2.1, the benefits of the EHW architecture compared to ANN were introduced. In this section, more details will be given. The EHW architecture provides classification in *parallel* for all the different numbers. Each path consists of *two* layers of gates in the AND-OR unit. The Selector could be implemented with a *one* gate layer. The Counter could effectively be implemented as a tree of gates. The Max Detector would consist of a comparator structure. Thus, all can be implemented in combinational logic. However, to improve the speed if necessary, registers could be introduced in the architecture together with pipelined processing. Thus, the architecture should be able to process one input to the array for *each* clock cycle. A neural network would typically have 32 inputs, 32 hidden units, and 6 outputs. This would result in at least $(32 \times 32 + 32 \times 6 = 1216)$ multiply-accumulate operations and $(32 + 6)$ sigmoid function operations. This would normally have to be executed as serial floating point operations on a processor. This seems to be a waste of resources since the input would still be binary values.

18.6. Results

This section reports the results from the experimental work undertaken on sign number classification. The evolutionary algorithm and parameters were the same as for the experiments presented in Section 18.4. A database of 198 images from traffic situations were used in the experiments. 115 contained a speed limit sign and 83 contained other signs or no sign at all. Many of the images were in various ways “difficult” (different brightness on sign, faded color on sign, etc.). The results were as follows (before number recognition was undertaken).

- (i) Is there a speed limit sign? A speed limit sign was found in 100 of the 115 images (87%). In those not found, the system stated that a speed limit sign was not present in the image.
- (ii) 78 of the 83 images without a speed limit were correctly refused (94%). Thus, only five images were sent to the final sign number recognition.

For *all* the 100 images (P) that the system correctly detected, a speed limit sign, the extracted number arrays were applied for evolving the hardware to perform classification. The number of arrays for the different speed limits is given in Table 18.3. The values for fitness scaling (see (18.5)) are also included. They are computed according to the following equation:

$$s = \frac{P}{P_n} \cdot k = \frac{100}{P_n} \cdot 0.7 = \frac{70}{P_n}, \quad (18.7)$$

TABLE 18.3. The number of extracted arrays classified for each speed limit.

Speed limit	Number of arrays (P_n)	Fitness scaling factor (s)
30	4	18
40	11	7
50	32	2
60	35	2
70	12	7
80	6	12

TABLE 18.4. The correct number classification performance in %.

Type of system	Step of evol.	Min	Max	Average
A: Performance of 16 outputs	1	67.6	75.7	71.44
B: Performance of 32 outputs	1	71.8	90.9	79.2
C: Performance of 32 outputs	2	92.9	96.0	94.5

TABLE 18.5. Performance of the best classifier architecture evolved.

Speed limit	30	40	50	60	70	80
Performance (in %)	100	100	84.4	91.4	100	100

P_n is the number of arrays for the given class (column two in the table) and k is a constant determining the ratio between the fitness from P_n training vectors (active) and $(P - P_n)$ training vectors (not active). The value of k was determined by experiments.

Table 18.4 summarizes the classification performance. All the experiments are based on a “fitness measure” equal to 16. In this table, “A” shows the performance when only the 16 outputs applied in the fitness function are applied in computing the performance (after step 1 evolution is finished). However, we see that it is better to apply all 32 outputs (“B”) since the average performance is about 8% higher. Thus, 16 OR gates with random input connections improve the performance. We see the importance of the step 2 evolution in “C” in which the performance is substantially increased. The average performance is more than 15% higher than for “B.” The best classifier provided a performance of 96%.

The performance for each speed limit for the best classifier is shown in Table 18.5. Only two speed limits have less than 100% correct classification.

In step 1 evolution, each AND-OR unit was evolved for 25,000 generations. Step 2 evolution needed less than 500 generations before the performance stopped increasing. We have not yet studied the performance of the 5 images that were wrongly sent to the number recognizer. However, these could probably be eliminated by a combination of matching on the “0” number and applying road grammar (as explained in Section 18.5.1). We have not fully explored all possible parameter settings for the architecture. Thus, there is a potential for further improving the performance. Further, a threshold unit could be introduced to avoid misclassifying numbers by rather indicating no distinct match.

In addition to achieving a high rate of correct classification, we have made much effort at the same time to reduce the processing time. This would be highly needed in a future real-time system. By applying digital logic gates for the number classification, we have almost eliminated the processing time for this operation. To have a robust and reliable system, more images should be analyzed and this is a part of our future work.

With the promising results so far, future work also consists of further improving the algorithms and implementing the most computational demanding parts in special hardware. More image data should be applied to better test the generalization performance. Further, possible inclusion of evolution in other parts of the system is of interest as well. Finally, a prototype to be applied in a vehicle will be implemented.

18.7. Conclusions

In this chapter, an EHW architecture for pattern classification including incremental evolution has been introduced. Experiments show that the performance can be substantially increased by applying incremental evolution compared to evolving a system directly in one operation. This has been shown for both prosthetic hand control and road sign image recognition. Another benefit is that the hardware would be compact and provide a cost effective platform.

The architecture would be appropriate for any application having real-time constraints and which could benefit from being online adaptable. The results illustrate that this is a promising approach for evolving systems for complex real-world applications.

Acknowledgment

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

Bibliography

- [1] E. Cantu-Paz, "A survey of parallel genetic algorithms," *Calculateurs Paralleles, Reseaux et Systems Repartis*, vol. 10, no. 2, pp. 141–171, 1998.
- [2] S. Fuji, "Development of prosthetic hand using adaptable control method for human characteristics," in *Proceedings of the 5th International Conference on Intelligent Autonomous Systems (IAS '98)*, pp. 360–367, Sapporo, Japan, June 1998.
- [3] D. Goldberg, *Genetic Algorithms in Search, Optimization, and Machine Learning*, Addison-Wesley, Reading, Mass, USA, 1989.
- [4] P. C. Haddow and G. Tufte, "An evolvable hardware FPGA for adaptive hardware," in *Proceedings of the Congress on Evolutionary Computation*, vol. 1, pp. 553–560, La Jolla, Calif, USA, July 2000.
- [5] W. D. Hillis, "Co-evolving parasites improve simulated evolution as an optimization procedure," *Physica D*, vol. 42, no. 1–3, pp. 228–234, 1990.
- [6] M. Iwata, I. Kajitani, H. Yamada, H. Iba, and T. Higuchi, "A pattern recognition system using evolvable hardware," in *Proceedings of the 4th International Conference on Parallel Problem Solving*

- from *Nature (PPSN '96)*, vol. 1141 of *Lecture Notes in Computer Science*, pp. 761–770, Springer, Berlin, Germany, September 1996.
- [7] I. Kajitani, T. Hoshino, N. Kajihara, M. Iwata, and T. Higuchi, “An evolvable hardware chip and its application as a multi-function prosthetic hand controller,” in *Proceedings of the 16th National Conference on Artificial Intelligence and the 11th Innovative Applications of Artificial Intelligence Conference (AAAI/IAAI '99)*, pp. 182–187, Orlando, Fla, USA, July 1999.
 - [8] M. Kimura, *The Neutral Theory of Molecular Evolution*, Cambridge University Press, Cambridge, UK, 1983.
 - [9] J. R. Koza, *Genetic Programming II: Automatic Discovery of Reusable Programs*, MIT Press, Cambridge, Mass, USA, 1994.
 - [10] W.-P. Lee, J. Hallam, and H. H. Lund, “Learning complex robot behaviours by evolutionary computing with task decomposition,” in *Proceedings of the 6th European Workshop on Learning Robots (EWLR '97)*, A. Birk and J. Demiris, Eds., vol. 1545 of *Lecture Notes in Computer Science*, pp. 155–172, Springer, Brighton, UK, August 1997.
 - [11] M. Murakawa, S. Yoshizawa, I. Kajitani, T. Furuya, M. Iwata, and T. Higuchi, “Hardware evolution at function level,” in *Proceedings of the 4th International Conference on Parallel Problem Solving from Nature (PPSN '96)*, vol. 1141 of *Lecture Notes in Computer Science*, pp. 62–71, Springer, Berlin, Germany, September 1996.
 - [12] R. N. Scott and P. A. Parker, “Myoelectric prostheses: state of the art,” *Journal of Medical Engineering & Technology*, vol. 12, no. 4, pp. 143–151, 1988.
 - [13] L. Sekanina and J. Torresen, “Detection of Norwegian speed limit signs,” in *Proceedings of the 16th European Simulation Multiconference on Modelling and Simulation (ESM '02)*, pp. 337–340, SCS Europe, Darmstadt, Germany, June 2002.
 - [14] T. Smith, P. Husbands, P. Layzell, and M. O’Shea, “Fitness landscapes and evolvability,” *Evolutionary Computation*, vol. 10, no. 1, pp. 1–34, 2002.
 - [15] R. Thomas, “Less is more [intelligent speed adaptation for road vehicles],” *IEE Review*, vol. 49, no. 5, pp. 40–43, 2003.
 - [16] J. Torresen, “A divide-and-conquer approach to evolvable hardware,” in *Proceedings of the 2nd International Conference on Evolvable Systems: From Biology to Hardware (ICES '98)*, M. Sipper, D. Mange, and A. Pérez-Uribe, Eds., vol. 1478 of *Lecture Notes in Computer Science*, pp. 57–65, Springer, Lausanne, Switzerland, September 1998.
 - [17] J. Torresen, “Scalable evolvable hardware applied to road image recognition,” in *Proceedings of the 2nd NASA/DoD Workshop on Evolvable Hardware (EH '00)*, J. Lohn, A. Stoica, D. Keymeulen, and S. Colombano, Eds., pp. 245–252, IEEE Computer Society, Palo Alto, Calif, USA, July 2000.
 - [18] J. Torresen, “Two-step incremental evolution of a digital logic gate based prosthetic hand controller,” in *Proceedings of the 4th International Conference on Evolvable Systems: From Biology to Hardware (ICES '01)*, vol. 2210 of *Lecture Notes in Computer Science*, pp. 1–13, Springer, Tokyo, Japan, October 2001.
 - [19] J. Torresen, J. W. Bakke, and L. Sekanina, “Efficient recognition of speed limit signs,” in *Proceedings of the 7th IEEE International Conference on Intelligent Transportation Systems (ITSC '04)*, pp. 652–656, Washington, DC, USA, October 2004.
 - [20] J. Torresen, J. W. Bakke, and L. Sekanina, “Recognizing speed limit sign numbers by evolvable hardware,” in *Proceedings of the 8th International Conference on Parallel Problem Solving from Nature (PPSN '04)*, vol. 3242 of *Lecture Notes in Computer Science*, pp. 682–691, Springer, Birmingham, UK, September 2004.
 - [21] J. Torresen and V. E. Skaugen, “A signal processing architecture based on RAM technology,” in *Proceedings of the 16th European Simulation Multiconference on Modelling and Simulation (ESM '02)*, pp. 317–319, SCS Europe, Darmstadt, Germany, June 2002.
 - [22] J. Torresen, “A scalable approach to evolvable hardware,” *Genetic Programming and Evolvable Machines*, vol. 3, no. 3, pp. 259–282, 2002.
 - [23] X. Yao and T. Higuchi, “Promises and challenges of evolvable hardware,” in *Proceedings of the 1st International Conference on Evolvable Systems: From Biology to Hardware (ICES '96)*, T. Higuchi, M. Iwata, and W. Liu, Eds., vol. 1259 of *Lecture Notes in Computer Science*, pp. 55–78, Springer, Tsukuba, Japan, October 1997.

- [24] M. Yasunaga, T. Nakamura, I. Yoshihara, and J. H. Kim, “Genetic algorithm-based methodology for pattern recognition hardware,” in *Proceedings of the 3rd International Conference on Evolvable Systems: From Biology to Hardware (ICES '00)*, J. F. Miller, A. Thompson, P. Thomson, and T. C. Fogarty, Eds., vol. 1801 of *Lecture Notes in Computer Science*, pp. 264–273, Springer, Edinburgh, UK, April 2000.

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

19

Evolutionary photogrammetric network design

Gustavo Olague and Enrique Dunn

19.1. Introduction

This chapter is about the automation of photogrammetric networks by means of evolutionary computation. The optimal design problem in photogrammetry may be stated: *given* the required quality (precision and reliability) of a set of parameters that are to be estimated, *find* the set of measurements that will achieve this with the least cost. This problem is known as photogrammetric network design (PND). Today, it is widely accepted that the process of designing a photogrammetric network should be done through simulation using an optimization approach. A truly optimal network would achieve the highest possible precision and reliability in the most economical manner! However, this turns to be difficult if not impossible in terms of design costs. Therefore, photogrammetrists have referred to network optimization as the search for a satisfactory configuration. In this way, the design process is concluded once a network achieves the accuracy within the limit of cost and/or time.

A first benefit of simulating photogrammetric networks through evolutionary computation is the natural interaction between both research areas based on the common goal of obtaining a suitable (satisfactory) solution (configuration). It is known that solving photogrammetric networks through a unique sequence of mathematical steps, possibly involving iteration, is not possible [19]. However, analytical design is applied to describe the criteria that are employed within the optimization approach. Moreover, a significant advantage of using evolutionary search lies in the gain of flexibility and adaptability to solve the task at hand, due to the combination of robust performance and global search characteristics. In this way, evolutionary computation should be understood as a general adaptable concept for problem solving, especially well suited for solving difficult optimization problems such as photogrammetric network design, instead of a collection of related and ready-to-use algorithms. On the other hand, from the evolutionary computation standpoint researchers are looking for challenging new problems



FIGURE 19.1. Typical sensor planning research is carried out with a robot manipulator and a CCD camera in what is known as a hand-eye configuration.

to enhance their understanding of evolutionary algorithms. In this avenue, PND represents a rich research subject.

Although the level of automation achieved in digital close-range photogrammetry has allowed the effective integration of vision metrology systems into industrial environments [16], the PND remains as an open problem in photogrammetry. Today, there are photogrammetric systems capable to achieve routinely accuracies above 1 : 100 000 of the principal dimension of the object [15]. However, to achieve such high accuracy in metrology, it is necessary to be a photogrammetric expert. Expertise is involved to decide about the arrangement and number of cameras to gain the strongest network, the appropriate placement of targets on the object to be measured, setting the lighting to obtain strongly contrasting targeted features, estimation of camera calibration and bundle adjustment, identification, and labeling of targets using feature detection techniques up to subpixel accuracy. Finally, the expert should run a program with all the above information and decide if the results are within some predetermined limit, or make some change, check for errors, and repeat it with new values until the objectives are achieved. The motivation of this research is to reduce the cost of vision system design and to equip autonomous inspection systems with photogrammetric network capabilities, for example, measurement robots used in flexible manufacturing, see Figure 19.1.

Nowadays, vision metrology is actively used in conjunction with computer vision, robotics, computer graphics, and other visual-related disciplines to solve problems in areas such as medicine, biology, architecture, industrial engineering, aerospace technology, to mention but a few. Traditionally, vision metrology has been enhanced also with methods from artificial intelligence and more recently with computational intelligence approaches. This research is about how evolutionary computation can be applied to design a photogrammetric network for industrial inspection tasks following a simulation approach. Computer simulation of close-range photogrammetric networks has been successfully employed and, with the sophistication of computers, a considerable boost to interactive network design has been achieved. The process of photogrammetric network design optimization through computer simulation can follow a number of approaches. One traditional procedure is based on the widely accepted classification scheme of

Grafarend [18], in which network design has been divided into four design stages of which only the first three are used in close-range photogrammetry [14].

- (i) *Zero-order design (ZOD)*: this stage attempts to define an optimal datum in order to obtain accurate object point coordinates and exterior orientation parameters.
- (ii) *First-order design (FOD)*: this stage involves defining an optimal imaging geometry which, in turn, determines the accuracy of the system.
- (iii) *Second-order design (SOD)*: this stage is concerned with adopting a suitable measurement precision for the image coordinates. It consists usually in taking multiple images from each camera station.
- (iv) *Third-order design (TOD)*: this stage deals with the improvement of a network through the inclusion of additional points in a weak region.

The initial step is to achieve a suitable observation and measuring scheme (FOD stage) in order to fulfill the required triangulation precision criteria. This entails the selection of an appropriate camera format, focal length, and image measurement system, as well as a first approximation to a suitable network geometry. FOD or SOD steps are applied if the network fails to achieve the criteria. If both corrections are insufficient, a completely new network needs to be proposed and the whole process is repeated iteratively. This chapter describes an evolutionary computation-based methodology for solving the fundamental stage in network design, that is, configuring an optimal imaging geometry. The problem is posed in terms of a global optimization design, which is capable of managing the problem using an adaptive strategy. The solution space is explored using both noncontinuous optimization and combinatorial search. Basically, the approach is to minimize the uncertainty of three-dimensional measurements using as a criterion the average variance of the 3D object points, presuming that the optimization satisfies a number of primary constraints and design decisions. This chapter details the optimization process based on an evolutionary strategy, and how the primary constraints and design decisions are managed in order to overcome the computational burden.

19.2. Problem statement

A main research problem on photogrammetric network design is devoted to the spatial distribution planning of cameras and targets in order to perform photogrammetric tasks. This section presents the modeling of a multistation sensor configuration in order to model accurate positions of 3D target points. Target points will be represented by error ellipsoids describing the uncertainty of their position. Changing the camera attitudes, position and orientation, changes the orientation and size of the error ellipsoids. The main question we will answer is where should we place the cameras in order to obtain the minimal 3D error?

19.2.1. Three-dimensional reconstruction

Brown [1] originally developed the bundle method in a fully general form. Today, the bundle method is recognized as a critical factor in exploiting the mensuration

potential of photogrammetry and is almost exclusively used in applications requiring high accuracy. However, nonrigorous approaches have been used to simplify the process of simulation. Here, we will describe a combined approach in which 3D reconstruction is achieved following computer vision and photogrammetric approaches. In computer vision, the camera network is commonly modeled from a geometric standpoint according to the projective model. This model is based on the fundamental assumption that the exposure center, the ground point, and its corresponding image point, all lie on a straight line.

Let u_{ij} and v_{ij} denote the photo coordinates of the image of point j in photograph i . For each pair of image coordinates $(u_{ij}, v_{ij})^t$ observed on each image, the following relationship exists:

$$\begin{aligned} u_{ij} &= \frac{m_{11}^i X_j + m_{12}^i Y_j + m_{13}^i Z_j + m_{14}^i}{m_{31}^i X_j + m_{32}^i Y_j + m_{33}^i Z_j + m_{34}^i}, \\ v_{ij} &= \frac{m_{21}^i X_j + m_{22}^i Y_j + m_{23}^i Z_j + m_{24}^i}{m_{31}^i X_j + m_{32}^i Y_j + m_{33}^i Z_j + m_{34}^i}. \end{aligned} \quad (19.1)$$

This system of equations assumes that light rays travel in straight lines, that all rays entering a camera lens system pass through a single point and that the lens system is distortion-less or, as is usual in highly accurate measurement, that distortion has been canceled out after having been estimated. In this way, a point in the scene P_j , $j = 1, \dots, n$, of homogeneous coordinates $(X_j, Y_j, Z_j, 1)^t$ is projected into points p_{ij} of image coordinates $(u_{ij}, v_{ij})^t$, through a projection matrix M_i , $i = 1, \dots, k$, of size 3×4 corresponding to the i th image. In this case, $k = 2$. Therefore, three-dimensional measurements can be obtained from several images. Each matrix M represents a mapping composed of a transformation $W \rightarrow C$ from the world coordinates W to the camera coordinates C given by

$$\begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} = \begin{bmatrix} R_{WC} & T_{WC} \\ 0_{1 \times 3} & 1 \end{bmatrix} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix}, \quad (19.2)$$

where the rotation matrix R_{WC} , which is a function of three rotation parameters (α, β, γ) and the translation vector T_{WC} , also of three degrees of freedom, characterizes the camera's orientation and position with respect to the world coordinate frame. Under perspective projection, the transformation from 3D-world coordinate system to the 2D-image coordinate is

$$\begin{pmatrix} su \\ sv \\ s \end{pmatrix} = \begin{bmatrix} R_{WC} & T_{WC} \\ 0_{1 \times 3} & 1 \end{bmatrix} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix}, \quad (19.3)$$

where the matrix

$$K = \begin{pmatrix} -k_u f & 0 & u_0 & 0 \\ 0 & k_v f & v_0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix}, \quad (19.4)$$

represents the intrinsic parameters of the camera, f is the focal length of the camera, (k_u, k_v) are the horizontal and vertical pixel sizes on the image plane, and (u_0, v_0) is the projection of the camera's center (principal point) on the image plane.

In computer vision, calibration is the process of estimating the intrinsic and extrinsic parameters of a camera. It can be thought of as a two-stage process in which we first compute the matrix M and then we compute the intrinsic and extrinsic parameters from M . Here, we follow the approach proposed by Faugeras and Toscani [11, 12] to calibrate each camera in order to obtain the 10 intrinsic and extrinsic parameters.

In photogrammetry, the bundle method accords simultaneous consideration to all sets (or "bundles") of photogrammetric rays from all cameras. The bundle method is based on a mathematical camera model comprised of separate functional and stochastic models. The functional model describing the relationship between the desired and measured quantities consists of the well-known collinearity equations. The collinearity equations, derived from the perspective transformation, are based on the same fundamental assumption of the projective model. Thus, the pinhole camera model (central projection) is the underlying model for both the projective approach and the collinearity-based approach. In this way, for each pair of image coordinates (x_{ij}, y_{ij}) observed on each image, the following pair of equations is written:

$$F_x = x_{ij} - x_p + f \frac{m_{11}(X_j - X_i^c) + m_{12}(Y_j - Y_i^c) + m_{13}(Z_j - Z_i^c)}{m_{31}(X_j - X_i^c) + m_{32}(Y_j - Y_i^c) + m_{33}(Z_j - Z_i^c)}, \quad (19.5)$$

$$F_y = y_{ij} - y_p + f \frac{m_{21}(X_j - X_i^c) + m_{22}(Y_j - Y_i^c) + m_{23}(Z_j - Z_i^c)}{m_{31}(X_j - X_i^c) + m_{32}(Y_j - Y_i^c) + m_{33}(Z_j - Z_i^c)},$$

where (x_{ij}, y_{ij}) denote the coordinates of point j on photograph i , f and (x_p, y_p) are the camera constant and image coordinates of the principal point of the sensor defining the sensor's orientation, (X_j, Y_j, Z_j) are the object space coordinates of the corresponding point feature, (X_i^c, Y_i^c, Z_i^c) are the object space coordinates of the perspective center, and m_{kl} are the elements of an orthogonal matrix which defines the rotation between the image and object coordinate systems. If each camera is calibrated following the Faugeras and Toscani approach, then it is possible to express the calibration through the collinearity equations. Thus, f is expressed as two projections along the main directions $\alpha_u = -K_u f$ and $\alpha_v = K_v f$. The image coordinates of the principal point are simply expressed as $x_p = u_0$ and $y_p = v_0$. In

this way, the analysis could be simplified after removing the intrinsic parameters to obtain the following models:

(a) computer vision

$$P' = \begin{bmatrix} R_{WC} & T_{WC} \\ 0_{1 \times 3} & 1 \end{bmatrix} P, \quad (19.6)$$

(b) photogrammetry

$$P' = R(P_j - P_c). \quad (19.7)$$

However, the last equation is expressed in nonhomogeneous coordinates. In order to state this equation as in the computer vision model, we write the equation as follows:

$$P' = RP_j - RP_c \quad (19.8)$$

to obtain

$$P' = \begin{bmatrix} R_{3 \times 3} & -RP_c \\ 0_{1 \times 3} & 1 \end{bmatrix} \begin{pmatrix} P_j \\ 1 \end{pmatrix}, \quad (19.9)$$

where $P_c = -R^t T_{WC}$.

19.2.2. 3D uncertainty estimation

Due to the nature of the measurement process, observations are accompanied by errors. Because of random errors, as evidenced by small differences between observations of the same quantity, observations can be regarded as random variables and their effects described by means of a stochastic model. Equation (19.5) can be linearized through the first-order development using the Taylor series. A functional model can be given as

$$\begin{aligned} \mathbf{v} &= \mathbf{Ay} - \mathbf{l}, \\ \mathbf{C}_1 &= \sigma_0^2 \mathbf{P}^{-1}, \end{aligned} \quad (19.10)$$

where \mathbf{l} , \mathbf{v} , and \mathbf{y} are vectors of observations, residuals, and unknown parameters, respectively; \mathbf{A} is the design or configuration matrix; \mathbf{C}_1 is the covariance matrix of observations; \mathbf{P} is the weight matrix; and σ_0^2 is the variance factor. In situations where \mathbf{A} is of full rank (i.e., where redundant or explicit minimal constraints are imposed), the parameter estimates $\hat{\mathbf{y}}$ and the corresponding cofactor matrix \mathbf{Q}_y and covariance matrix \mathbf{C}_y are obtained as

$$\hat{\mathbf{y}}(\mathbf{A}^t \mathbf{P} \mathbf{A})^{-1} \mathbf{A}^t \mathbf{P} \mathbf{l} = \mathbf{Q}_y \mathbf{A}^t \mathbf{P} \mathbf{l}, \quad (19.11)$$

$$\mathbf{C}_y = \sigma_0^2 \mathbf{Q}_y. \quad (19.12)$$

The ultimate aim of any photogrammetric measurement is the determination of triangulated object point coordinates along with estimates for their precision. The bundle method is simplified by considering two groups of parameters in the vector $\hat{y} : \hat{y}_1$, comprising exterior orientation (self-calibration parameters were not considered for simplicity), and \hat{y}_2 containing object coordinate corrections. Equation (19.11) then assumes the form

$$\begin{pmatrix} \hat{y}_1 \\ \hat{y}_2 \end{pmatrix} = \begin{pmatrix} \mathbf{A}_1^t \mathbf{P} \mathbf{A}_1 & \mathbf{A}_1^t \mathbf{P} \mathbf{A}_2 \\ \mathbf{A}_2^t \mathbf{P} \mathbf{A}_1 & \mathbf{A}_2^t \mathbf{P} \mathbf{A}_2 \end{pmatrix}^{-1} \begin{pmatrix} \mathbf{A}^t \mathbf{P} \mathbf{L} \\ \mathbf{A}^t \mathbf{P} \mathbf{L} \end{pmatrix} \quad (19.13)$$

and the cofactor matrix \mathbf{Q}_y can be written as

$$\mathbf{Q}_y = \begin{pmatrix} \mathbf{Q}_1 & \mathbf{Q}_{1,2} \\ \mathbf{Q}_{2,1} & \mathbf{Q}_2 \end{pmatrix}. \quad (19.14)$$

The design optimization goal for precision is to achieve an optimal form of \mathbf{Q}_2 and therefore the covariance matrix of object point coordinates (X_j, Y_j, Z_j) , considering the applicable design constraints. The criterion used in the minimization process is the average variance along the covariance matrix σ_c^2 , that is,

$$\sigma_c^2 = \frac{\sigma_0^2}{3n} (\text{trace } \mathbf{Q}_2). \quad (19.15)$$

Before dramatic improvements in computer processing power in recent years, a valid criticism of designing close-range networks by simulation was the computation time required for a bundle adjustment after each design-iteration even for relatively small networks. As shown by Brown [1], the covariance matrix can be obtained through the equation

$$\mathbf{Q}_2 = \sigma_0^2 [(\mathbf{A}_2^t \mathbf{P} \mathbf{A}_2)^{-1} + \mathbf{K}], \quad (19.16)$$

where

$$\begin{aligned} \mathbf{K} &= \mathbf{M} \mathbf{Q}_1 \mathbf{M}^t, \\ \mathbf{M} &= (\mathbf{A}_2^t \mathbf{P} \mathbf{A}_2)^{-1} \mathbf{A}_2^t \mathbf{P} \mathbf{A}_1. \end{aligned} \quad (19.17)$$

In this way, the determination of \mathbf{Q}_2 using this approach represents a rigorous approach that is termed *total error propagation (TEP)*. On the other hand, it has been demonstrated (Fraser [16]) that, for a wide range of convergent photogrammetric networks, $\mathbf{K} = \mathbf{0}$. This consideration is nonrigorous in that it implicitly assumes that exterior orientation parameters exhibit no dispersion and is called *limited error propagation (LEP)*. The perspective parameters are assumed to be error free and the variances in object point coordinates arise solely from the propagation of

random errors in the image coordinate measurements. What is remarkable from a network design standpoint is that, for strong networks (convergent networks), LEP is sufficiently accurate compared to TEP, causing considerable computing savings. Following the computer vision model, it is possible to apply the LEP approach in order to simplify the uncertainty analysis. The analytical method that is applied in our work takes account of the fact that the uncertainty of the image measurements propagates to the world measurements according to the following proposition.

Proposition 19.1. *Given a random variable $p \in \mathbb{R}^m$, of Gaussian distribution, mean $E[p]$, and covariance Λ_p , and $P \in \mathbb{R}^n$, the random vector given by $P = f(p)$, where f is a function of class C^1 , the mean of P can be approximated to a first-order Taylor expansion by $f(E[p])$ and its covariance by*

$$\Lambda_P = \frac{\partial f(E[p])}{\partial p} \Lambda_p \frac{\partial f(E[p])^t}{\partial p}. \quad (19.18)$$

19.3. Objectives and constraint's requirements in sensor planning for vision metrology

This section provides a broad list of objectives and constraints that could be found in sensor planning research in the case of passive sensors such as a CCD camera. The first set of constraints depends only on the optical lens design, and is defined by three fairly generic feature detectability constraints. These constraints require that the scene features, which are being imaged by the camera, are in focus, within the field of view of the camera, and within a sufficient resolution.

19.3.1. Optical constraints

The optical constraints characterize the distance between the camera viewpoint and the scene features of interest. In general, a synthesis approach has been taken to solve the problem. Each optical constraint is characterized by an analytical relationship. The admissible domain of sensor placement and settings is bounded by the frustum defined through the analytical relationships, and where a globally admissible viewpoint is sought.

(i) *Resolution constraint.* The resolution of the object in the image must be sufficient to support image mensuration to the desired precision. The minimum required scene feature resolution is a function of the camera optical lens, the pixel spacing of the sensor, and the camera location and orientation with respect to the feature of interest. For example, the pixel resolution of a given feature is increased when the camera is positioned closer to the observed feature while maintaining the same orientation, or the focal length of the lens is increased.

(ii) *Field of view constraint.* One of the major considerations of photogrammetrists in network design is to attempt to place each camera station such that the

entire scene lies within the field of view of the camera. This goal is motivated by the fact that capturing the entire scene in each frame will lead to a simplified design. However, this goal is contrary to the resolution constraint as long as a shorter-length lens could be less desirable from a precision point of view. Of course, the field of view goal is not always possible like in the case of complex 3D objects due to occlusions of the environment or self-occlusions produced by the object itself.

(iii) *Depth of field constraint.* Features to be measured should lie within the range of sufficiently sharp focus. Sharply focused images are desirable because they contain more information. Moreover, images that are not sharply focused have lost image contrast and detail and their edges are blurred. Depth of field is a function of the sensor's aperture and effective focal length, the admissible diameter of the circle of confusion, and the sensor-to-feature distance.

For any camera and lens setting, there is only one object distance called the focus distance, for which points are in perfect focus. While the focus distance is unique for a given setting of the lens, in practice there exists a range of object distances for which an image is considered to be focused. However, a reasonable relaxation of this limit can be tolerated when digital image mensuration techniques are employed. The depth of field remains as a factor constraining both image scale and sensor configuration geometry.

19.3.2. Environmental constraints

Planning an appropriate camera configuration needs to take into account a set of constraints pertaining to the environment. Next, we define three common environmental constraints.

(i) *Visibility constraint.* Obstructions in the workplace often preclude the possibility of imaging all points of interest which would otherwise lie within the field of view of the camera. Thus, the visibility of a feature of an object from a particular viewpoint could fail if the feature is occluded by either some part of the object on which it lies, self-occlusion, or other objects in the environment. This problem is produced by the opaqueness of the objects. Also, the visibility could fail if the feature lies outside the field-of-view limits of the sensor. These two cases of rendering a feature invisible are addressed as separate constraints in the literature. The first is named the visibility constraint, while the second is the field-of-view constraint. In vision metrology, the visibility constraint may affect the triangulation accuracy, because the occlusions could cause a poor contribution of the imaging geometry and convergent angles. Therefore, the strength of the intersection geometry may be compromised if the number of rays falls below that of a basic generic network.

(ii) *Workspace constraint.* The workspace in which the sensor planning survey is conducted can impose restrictions on the camera placement. The sensor configuration can be limited by the workspace if the environment imposes restrictions. Those restrictions could be produced by the walls of the room, any objects between the camera and the scene being studied, or the kinematic and dynamic constraints generated when the camera is placed by a robot manipulator. The workspace constraint limits the availability of viewpoints where a camera could be placed.

(iii) *Illumination constraint.* The quality of object illumination plays an important role in the quality of the sensing tasks. Therefore, sensor planning should also include planning of the illumination. A vision task is specified by an object model that includes photometric properties, light source type, as well as geometric properties of the feature object. For example, the light source is assumed to be a point light source, and no area light sources are supported. The background must be dark enough to distinguish the object clearly from the background. No secondary reflection or ambient light is considered. All faces have uniform reflectance; no colour or shading effect is considered. In summary, for natural object features such as edges, however, the illumination constraint is very critical. On the other hand, in photogrammetry retro-reflective targets are most commonly the features to be measured. Criteria for the illumination of these targets are well understood and can generally be satisfied by ring-flash illumination of the object from each camera station.

19.3.3. Photogrammetric constraints

Including the constraints that we have described just above, there are a number of constraints that are studied in vision metrology. The photogrammetric network design must deal with a series of constraints in order to select an optimal camera distribution. The accuracy of the system is related to the imaging geometry (main objective in PND) as well as the convergence angle of each camera with respect to each object surface. Considering all the constraints limiting the search space, we identify the following main objective and constraints.

(i) *Contribution to intersection angles or the imaging geometry.* The contribution of a single camera station to a network configuration cannot be treated in isolation. For example, at least 3 rays are necessary to ensure strong intersection angles at a target or feature. The stations or cameras rising these rays must be placed at positions separated in 3D space. Each camera adds strength to the ray intersection geometry, contributing to the triangulation precision. Consequently, each camera station cannot be independently added to a configuration, but rather consideration should be given to how it will interact with the other stations to produce strong multiray intersections. Hence, within a camera placement system the main objective is to know the contribution of each camera with respect to the others. Two fundamental questions need to be answered. How many cameras will be needed and where should they be placed? A solution to the second question has been proposed by [23]. If we have solved where to place a given number of cameras, it is possible to compute then the appropriate number of cameras for the task of measuring an object.

(ii) *Number and distribution of image points.* The statistical reliability of the photogrammetric orientation can be enhanced through the greater redundancy provided by extra targets. The number and distribution of image points also significantly influences the precision of recovery of sensor calibration parameters in self-calibrating bundle adjustment. However, there is a relative independence of triangulation precision and number of targets. The independence of target density

and object point precision is explained in terms of the limited error propagation, which takes into account only the intersection geometry at each point and the directional dispersion of the imaging rays to the point. On the other hand, it is important to attempt to maximize the total number of object points seen at a station since an increasing number of image points is accompanied by an improvement in the internal reliability of the individual observations.

(iii) *Incidence angle constraint or convergence angle.* The reliability of image measurements from directions close to coplanar is difficult and even impossible to obtain. The minimum allowable incidence angle is dependent on the type of feature, its geometry, and its material, as well as the illumination. The accuracy of the measurement with respect to the convergence angle is a function of the viewing direction and the surface normal at the feature. In the case of circular targets, the minimum convergence angle is about 20 to 30 degrees for the kind of retroreflective targets that are normally used. An hyperbolic model describing the behavior of the error measurement for this kind of targets has been proposed recently, see [23].

19.4. An evolutionary photogrammetric network design system

As discussed in previous sections, the PND problem offers a geometrical design problem with an intricate ensemble of mathematical, optical and operational considerations. These aspects can be addressed in a coherent manner by approaching the PND problem in optimization terms. However, an optimization needs to be carried out over a poorly understood search space that presents numerical and combinatorial aspects. A close examination of the problem will yield the following challenges.

(i) The search space is nonlinear. The relationship between the imaging geometry of multiple cameras and reconstruction accuracy of a 3D object is modeled by a complex mathematical model that is difficult to address by analytical or numerical means.

(ii) The search space offers discontinuities. Concave 3D objects present the problem of self-occlusion for visual sensing, giving rise to a combinatorial optimization process.

(iii) The search space is multimodal. There exist multiple configurations (with very different geometrical topology) which provide very similar results.

EC techniques offer a powerful paradigm to address all these issues, due to their stochastic and population-based metaheuristics. Indeed, the flexibility of the EC paradigm allows us to extend the study of PND beyond a purely geometrical problem through the use of a properly designed simulation based approach. The evolving positions of cameras (EPOCA) system is an example of a system developed following these principles.

EPOCA is a CAD-based planning system which incorporates automated camera network design algorithms with a 3D simulation environment and interface. The system takes as input a 3D CAD model of the object and determines automatically the location and attitude of a set of cameras converging on the object.

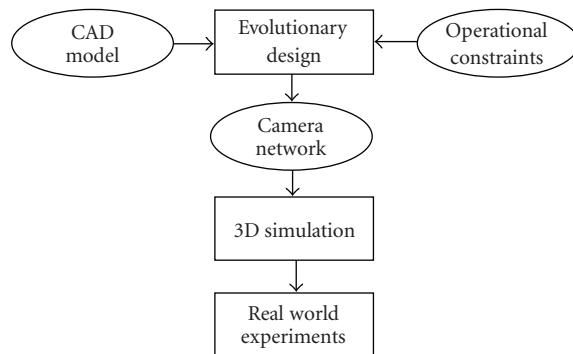


FIGURE 19.2. Outline of evolutionary photogrammetric network design.

The ongoing development of EPOCA has extended this basic scenario to include the operation of an active vision system as well as to address different evolutionary optimization techniques. The following sections describe some of the different ways in which EC has been incorporated into the EPOCA system to address different issues in the PND problem.

19.5. Designing an optimal camera network for 3D reconstruction

Evolutionary algorithms operate over a set of parameterized solutions through a set of stochastic heuristic functions. The evolution process optimizes a photogrammetric network design criterion under a number of constraints, see Section 19.2. In this way, there is a special nomenclature used in the EC literature for the basic algorithmic elements used in the development of an evolutionary algorithm. For example, the set of solutions is called a population while each single solution is called an individual. The set of stochastic functions are called genetic operators while the criterion to optimize is termed a fitness function. Accordingly, the goal of the EPOCA system is to evolve a population of camera configurations in order to solve the PND problem. The first algorithmic aspect to be addressed is the definition of a suitable genetic representation (see Figure 19.3).

19.5.1. Genetic representation

The design of an optimal camera network is considered as the problem of determining an optimal imaging geometry. In turn, the main issues to be addressed are the convergence angle of each camera with regard to the object features, as well as the relative orientation of each camera with respect to the other sensing stations. Our evolutionary approach requires a parameterization of the PND that effectively represents these aspects. The viewing sphere model is adopted in order to have a representation that provides 3D convergent networks while maintaining a parameterization of reduced dimensionality. Under this viewing model all cameras are

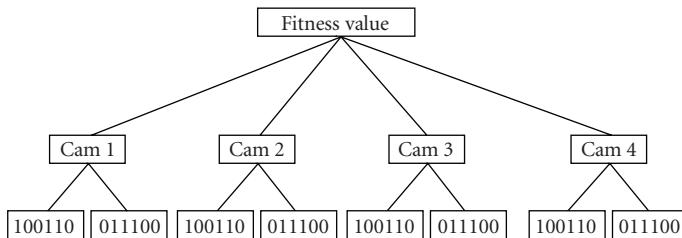


FIGURE 19.3. Genetic representation of an individual. The camera positions are encoded with a binary string using the hierarchical structure of a tree as in the linear genetic programming.

placed on the surface of a sphere which is centered on the object under observation. In this way, the position and orientation of a single camera can be expressed by the spherical coordinates on the viewing sphere, denoted by (α, β) . The EPOCA system uses a binary encoding of these coordinates which allows the system to effectively deal with the combinatorial aspects of our optimization problem. In this way, given a coding size L (i.e., number of bits used to represent each variable), a camera network of n cameras can be represented by a binary vector

$$X \in \mathbb{B}^{2Ln} \quad \text{where } x_i \in \{0, 1\} \text{ for } i = 1, \dots, n. \quad (19.19)$$

Accordingly, each camera network is represented by a binary vector

$$\begin{aligned} \alpha_i &= \{x_{L(2i-2)+1}, x_{L(2i-2)+2}, \dots, x_{L(2i-2)+L}\}, \\ \beta_i &= \{x_{L(2i-1)+1}, x_{L(2i-1)+2}, \dots, x_{L(2i-1)+L}\} \quad \text{for } i = 1, \dots, n. \end{aligned} \quad (19.20)$$

Moreover, the viewing sphere model allows a straight-forward incorporation of the optical parameters. This simplification can be achieved by a priori determination of the optical constraints (field of view, focus, and resolution) allowing each camera to observe the complete object. The optical constraints are satisfied once the focal length and distance to the object are properly selected.

19.5.1.1. Recombination and mutation

In genetic algorithms, the search for improved solutions is driven by the process of recombining individuals from the population. This recombination is implemented through a genetic operator called crossover. For a binary representation, the role of this operator consists in combining the information of two individuals in the population in order to generate two new individuals. This is normally done by selecting a single bit position and exchanging the information on either side of this “cut” position. In our algorithm, this heuristic is implemented at the camera level. In this way, a single parameter is combined only with the corresponding parameter in another individual. Hence, given two binary vectors representing a camera parameter, $X, Y \in \mathbb{B}^L$, as well as a cut position $1 \leq c \leq L$, the two new

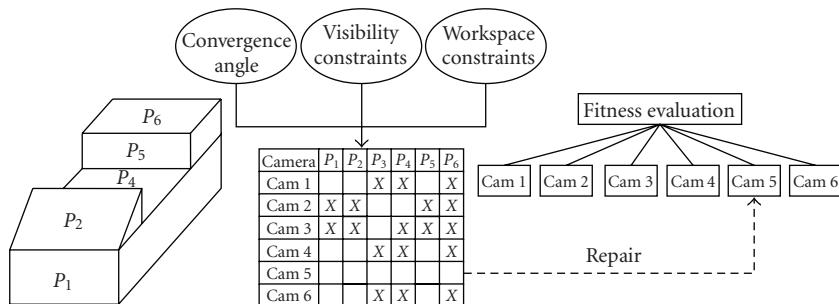


FIGURE 19.4. Integrating different constraints on PND. The object being observed is divided into a number of disjoint regions. The sensing contribution of each camera position is validated for each object region. Invalid sensing positions are automatically repaired.

binary vectors X' , Y' will be obtained as follows:

$$X'_i = \begin{cases} X_i & \text{if } i < c, \\ Y_i & \text{otherwise,} \end{cases} \quad Y'_i = \begin{cases} Y_i & \text{if } i < c, \\ X_i & \text{otherwise,} \end{cases} \quad \text{for } i = 1, \dots, L. \quad (19.21)$$

Mutation is also incorporated into our approach by stochastically changing a single bit position subject to a predetermined probability.

19.5.2. Constraint handling

A valid camera configuration is one that provides sufficient image measurements to effectively reconstruct all the object features under study. Moreover, such configuration must comply with the different constraints mentioned in previous sections. This requires evaluating different types of constraints for a single camera as well as for the complete camera network (see Figure 19.4).

Visibility constraints are incorporated into the EPOCA system by means of an offline ray-tracing module that evaluates the visibility of each object feature from the different camera positions available in the viewing sphere. The results of each visibility evaluation are stored in a database for online query during the evolution process. It is required that every camera observes a nonempty subset of the object features. However, depending on the object being studied, it is possible for the evolution process to produce invalid camera positions. Such positions are validated and corrected if necessary.

The incidence angle constraint is enforced during fitness evaluation. For a single feature, only those cameras that comply with this constraint contribute to the feature reconstruction process. A closely related constraint is the lack of redundancy in image measurements due to an insufficient sampling. EPOCA validates the last constraint and penalizes the invalid configurations assigning a predetermined fitness value. These constraints are stored in an internal database that keeps record of which camera positions can observe a given object region.

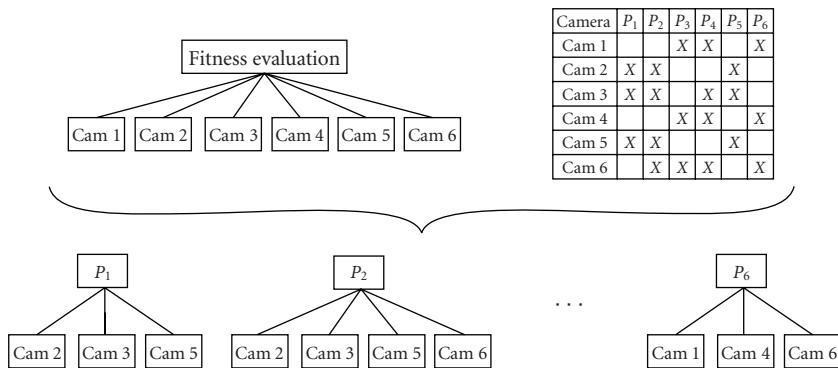


FIGURE 19.5. Fitness evaluation of an individual. A mathematical model is dynamically formulated for each object region. This is based on a particular camera configuration value and the constraint database previously calculated.

19.5.3. Evaluation

In our evolutionary algorithm, the evaluation of an individual is given by the quality of the reconstruction (in terms of precision) derived from the camera configuration. The uncertainty of our 3D reconstruction can be evaluated using the different criteria presented in previous sections. In order to attain a reasonable computational time for our evolutionary algorithm, the approach based on the error propagation phenomena was originally developed [23]. The evaluation process partitions the objects into separate regions of interest. The mathematical model for 3D reconstruction is built for each object region considering the subset of cameras observing it. In this way, the fitness evaluation is a dynamical process that depends on the topology of the object, as well as on the evaluated camera network (see Figure 19.5).

19.5.4. Experimentation

A number of experiments were carried out in order to validate the proposed approach. All the experiments used a population of 30 individuals with a coding length $L = 10$. Hence, for a network of n cameras, each individual can be represented by a vector of $2 \times 10 \times n$ binary values. For large networks this would be a very difficult optimization task. However, thanks to the tree-based representation adopted, the search space is efficiently explored by our evolutionary algorithm. Figure 19.6 illustrates the type of results provided by the system. The obtained networks are good in terms of camera distribution and ray inclination. Moreover, for the case of a planar structure, the results are consistent with networks previously proposed by photogrammetrists [13]. In this way, human competitive results can be obtained through the use of evolutionary techniques for the solution of a complex design problem.

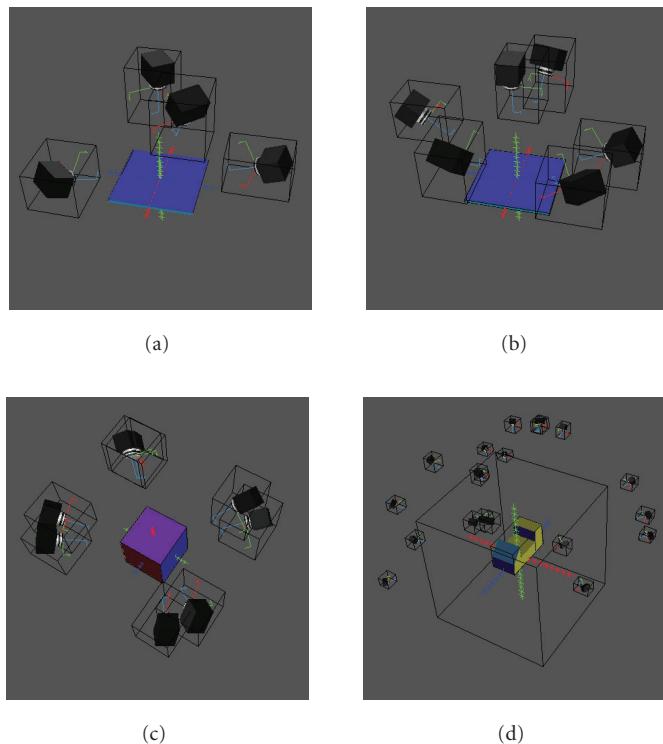


FIGURE 19.6. Experimental results. Examples of camera networks designed by the EPOCA system; (a) four cameras observing a plane; (b) six cameras observing a plane; (c) eight cameras observing four planes; (d) twenty cameras observing a complex object. Each of these configurations correspond to the best results that were obtained in several executions.

19.6. Incorporating a manipulator robot

Once an appropriate imaging geometry has been achieved, the next phase is to consider the manner by which the sensing actions will be carried out. Here, the case of a manipulator robot used for automated image acquisition is considered. A straight forward approach would be to directly use the optimal configuration designed by EPOCA. However, a more general approach is to include the different operational aspects of the photogrammetric project into the design of an optimal imaging geometry. This requires to consider several aspects such as the robotic system and computational costs, as well as their relationship with the main task of accurate reconstruction. Stating the PND problem in multiobjective (MO) optimization terms provides a general framework where all these aspects can be studied. Hence, extending the EPOCA system to include this framework allows for a better characterization of the different tradeoffs involved in the overall performance of an automated photogrammetric system.

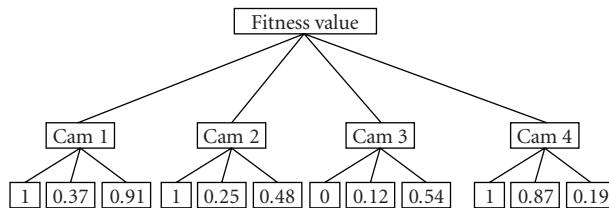


FIGURE 19.7. Modified genetic representation of an individual. The tree structure is extended to incorporate a “control bit” for each camera. In this example the third camera will be excluded since the corresponding control bit is turned off.

19.6.1. Genetic representation

The original representation of the EPOCA system was modified to real value encoding in order to study the fine-grain compromises between different objectives. In this way, for a network of n camera positions, the real-coded genotype utilized is given by

$$\vec{x} \in \mathbb{R}^{2n} \quad \text{where } \alpha_i = x_{2i-1}, \quad \beta_i = x_{2i} \quad \text{for } i = 1, \dots, n. \quad (19.22)$$

Furthermore, the principles presented in the *structured genetic algorithm* proposed in [3] are incorporated in our approach in order to evaluate networks of different sizes during the same evolution process (see Figure 19.7). Therefore, an additional binary section of the form

$$\vec{x}^b \in \mathbb{B}^n, \quad \text{where } x_i^b \in [0, 1] \text{ for } i = 1, \dots, n, \quad (19.23)$$

is added to the genotype. The value of each bit x_i^b determines the inclusion of a camera position into a network specification. Thus, our extended genotype is of the form $X = \langle \vec{x}^b, \vec{x} \rangle$.

19.6.2. Recombination and mutation

The crossover and mutation operators used for real-valued encoding are different from those used for the binary representation. The simulated binary crossover (SBX) emulates the working principle of the single point crossover operator on binary strings. From two parent solutions P_1 and P_2 , it creates two children C_1 and C_2 as follows:

$$\begin{aligned} C_1 &= 0.5[(1 + \beta)P_1 + (1 - \beta)P_2] & \text{with } \beta = \begin{cases} (2u)^{1/(\eta_x+1)} & \text{if } u < 0.5, \\ \left(\frac{1}{2(1-u)}\right)^{1/(\eta_x+1)} & \text{otherwise.} \end{cases} \\ C_2 &= 0.5[(1 - \beta)P_1 + (1 + \beta)P_2] \end{aligned} \quad (19.24)$$

The spread factor β depends on a random variable $u \in [0, 1]$ and on a user-defined nonnegative value η_x that characterizes the distribution of the children in relation to their parents. The mutation operation modifies a parent P into a child C using the boundary values $P^{(\text{LOW})}$ and $P^{(\text{UP})}$ of each of the decision variables in the following manner:

$$C = P + (P^{(\text{UP})} - P^{(\text{LOW})})\delta \quad \text{with } \delta = \begin{cases} (2u)^{1/(\eta_m+1)} - 1 & \text{if } u < 0.5, \\ 1 - [2(1-u)]^{1/(\eta_m+1)} & \text{otherwise.} \end{cases} \quad (19.25)$$

19.6.3. Constraint handling

The integration of a robotic arm into the image acquisition system requires the consideration of the kinematic restrictions of the manipulator since they can limit the space of attainable camera positions. These restrictions are evaluated offline, through a simulation environment that solves the inverse kinematic problem, and stored in a database for online query.

19.6.4. Evaluation

The criterion based on the error propagation phenomena is used for evaluating the precision of 3D measurements. The operational cost of robot displacement is evaluated through a 3D instance of the traveling salesman problem. For camera networks of small size an exhaustive search is carried out, while larger networks use a greedy heuristic combined with a 2-opt algorithm. Furthermore, computational cost is considered as a function of the number of cameras used for reconstruction. In this case, the fitness of an individual depends on its Pareto ranking with respect to the population. Accordingly, selection pressure will promote individuals which offer an optimal tradeoff between objectives which are known as Pareto optimal solutions.

19.6.5. Experimentation

Experimentation was carried out for the simulation of a complex three-dimensional object under observation by a manipulator robot. The goal was to obtain a photogrammetric network that is optimal in terms of reconstruction accuracy and manipulator motion (see Figure 19.8). It is reasonable to assume that the most *efficient* configuration, in terms of motion alone, will be one where the robot takes images from virtually the same viewpoint. On the other hand, an optimal network in terms only of precision will be one where a well-distributed set of viewpoints is achieved. Using a population of $N = 100$ individuals, the evolutionary algorithm was executed for a period of $t_{\text{max}} = 100$ generations. The crossover probability was $P_x = 0.8$ with a spread factor $\eta_x = 2$. The mutation probability was $P_m = 0.012$ with $\eta_m = 2$. The resulting set of optimal tradeoffs is illustrated in Figure 19.9. The horizontal axis reflects the magnitude of the 3D reconstruction uncertainty.

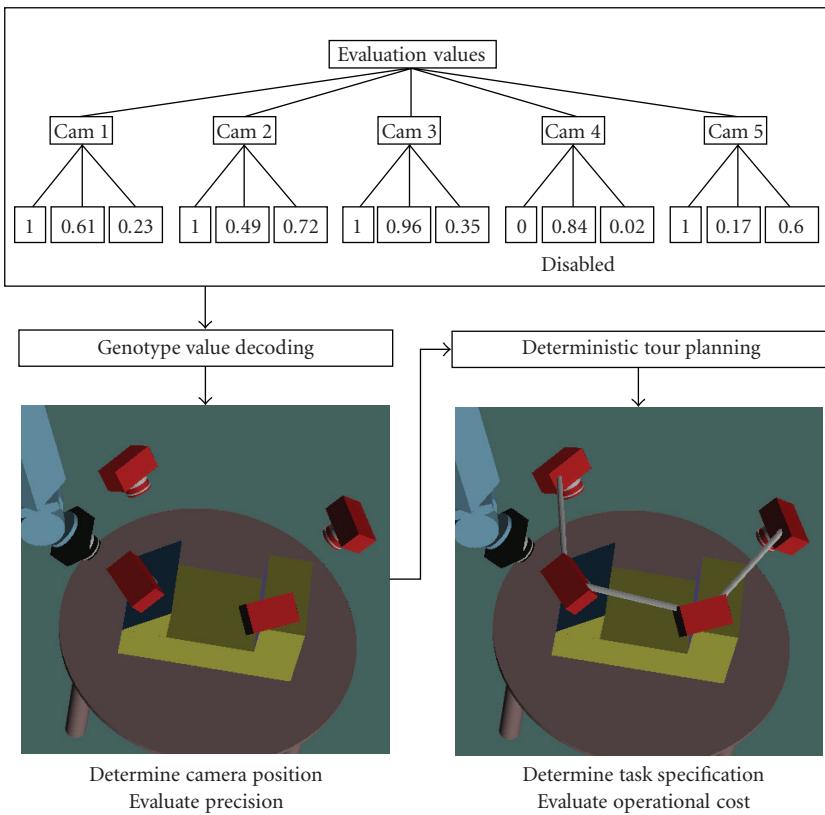


FIGURE 19.8. Genotype to phenotype transformation. A genotype representation that allows for up to five cameras is depicted. Only four are expressed into the decoded camera network, from which a robot tour is determined.

The vertical axis illustrates the total distance traveled by the manipulator. A single point in this plot represents the function values corresponding to a given sensing specification. The asymptotic behavior on both axes, as well as the convex shape of the Pareto front, support the assumption that our sensor planning problem is indeed multiobjective. These results provide an unbiased representation of the different options available for selecting a camera configuration in an automated environment. Obviously, a selection must be made from this set of possible solutions. However, such a decision is dependent on the needs of a particular application and/or the individual preferences of the practitioner.

19.7. Conclusions

In this chapter we have presented a solution to the problem of configuring an optimal photogrammetric network with the goal of achieving highly accurate 3D

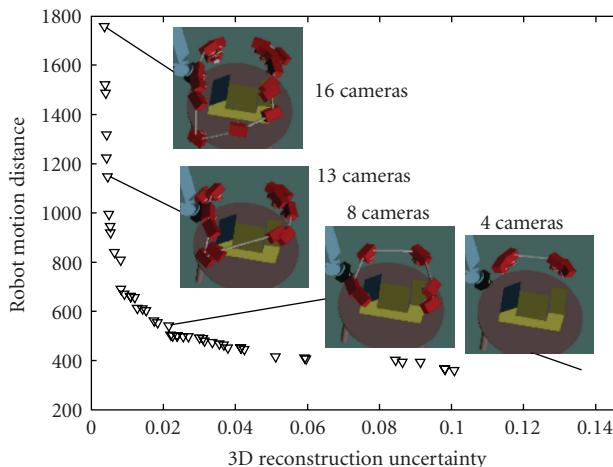


FIGURE 19.9. Experimental results. The set of optimal tradeoff solutions is illustrated along with some of their corresponding sensing specifications. Note the existence of networks with different sizes that were obtained during a single execution.

measurements in terms of an optimization design. This chapter has been divided into two main parts. The first was devoted to reviewing the bundle method and an analytical error model from which a criterion was derived. Also, we have reviewed several constraints that need to be taken into account such as the optical constraints and environmental constraints. The criterion we have chosen was the average variance extracted from the covariance matrix of the object point coordinates. This criterion was selected in order to simplify the stochastic optimization process. The second part was devoted to our approach based on evolutionary computation used to solve the PND problem in terms of a global optimization method in order to minimize the criterion. Due to the occlusions of points, caused by the different constraints, the problem presents discontinuities, which leads to combinatorial aspects in the optimization process. These constraints are logically incorporated into the evolutionary computation strategy. The strategy proposed in this work provides the ability to adapt the PND to produce a suitable multistation configuration for each new inspection task. The optimization search strategy considers the large number of competing considerations towards the optimal satisfaction of all placement constraints until the best acceptable configuration is found. This work has shown how evolutionary computation is able to design human competitive photogrammetric networks. We believe that in the future the design of photogrammetric networks will be based on this kind of optimization-based approach.

Acknowledgment

This research was funded through the LAFMI (Laboratoire Franco-Mexicain d'Informatique) project sponsored by CONACyT-INRIA.

Bibliography

- [1] D. C. Brown, "A solution to the general problem of multiple station analytical stereo triangulation," *RCA Tech. Rep. 43 (AFMTC TR 58-8)*, RCA Missile Test Project, Patrick Air Force Base, Fla, USA, 1958.
- [2] C. Coello Coello, D. Van Veldhuizen, and G. Lamont, *Evolutionary Algorithms for Solving Multi-Objective Problems*, Kluwer Academic Publishers, Dordrecht, The Netherlands, 2002.
- [3] D. Dasgupta and D. R. McGregor, "sGA: a structured genetic algorithm," *Tech. Rep. IKBS-11-93*, Department of Computer Science, University of Strathclyde, Strathclyde, UK, 1993.
- [4] K. Deb, *Multi-Objective Optimization Using Evolutionary Algorithms*, John Wiley & Sons, New York, NY, USA, 2001.
- [5] K. Deb, S. Agrawal, A. Pratab, and T. Meyarivan, "A fast elitist non-dominated sorting genetic algorithm for multi-objective optimisation: NSGA-II," in *Proceedings of the 6th International Conference on Parallel Problem Solving from Nature (PPSN '00)*, vol. 1917 of *Lecture Notes in Computer Science*, pp. 849–858, Springer, Paris, France, September 2000.
- [6] E. Dunn and G. Olague, "Pareto optimal camera placement for automated visual inspection," in *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS '05)*, pp. 3821–3826, Edmonton, Alberta, Canada, August 2005.
- [7] E. Dunn and G. Olague, "Evolutionary computation for sensor planning: the task distribution plan," *EURASIP Journal on Applied Signal Processing*, vol. 2003, no. 8, pp. 748–756, 2003.
- [8] E. Dunn, G. Olague, and E. Lutton, "Automated photogrammetric network design using the parisian approach," in *Proceedings of the 7th European Workshop on Evolutionary Computation in Image Analysis and Signal Processing (EvoIASP '05)*, vol. 3449 of *Lecture Notes in Computer Science*, pp. 356–365, Springer, Lausanne, Switzerland, March–April 2005.
- [9] E. Dunn, G. Olague, E. Lutton, and M. Schoenauer, "Pareto optimal sensing strategies for an active vision system," in *Proceedings of the IEEE International Conference on Evolutionary Computation (CEC '04)*, vol. 1, pp. 457–463, Portland, Ore, USA, June 2004.
- [10] E. Dunn and G. Olague, "Multi-objective sensor planning for efficient and accurate object reconstruction," in *Proceedings of the 6th European Workshop on Evolutionary Computation in Image Analysis and Signal Processing (EvoIASP '04)*, vol. 3005 of *Lecture Notes in Computer Science*, pp. 312–321, Springer, Coimbra, Portugal, April 2004.
- [11] O. D. Faugeras and G. Toscani, "The calibration problem for stereo," in *Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR '86)*, pp. 15–20, Miami Beach, Fla, USA, June 1986.
- [12] O. D. Faugeras and G. Toscani, "Camera calibration for 3D computer vision," in *Proceedings of International Workshop on Machine Vision and Machine Intelligence*, pp. 240–247, Tokyo, Japan, February 1987.
- [13] C. S. Fraser, "Optimization of precision in close-range photogrammetry," *Photogrammetric Engineering & Remote Sensing*, vol. 48, no. 4, pp. 561–570, 1982.
- [14] C. S. Fraser, "Network design considerations for non-topographic photogrammetry," *Photogrammetric Engineering & Remote Sensing*, vol. 50, no. 8, pp. 1115–1126, 1984.
- [15] C. S. Fraser, "Photogrammetric measurement to one part in a million," *Photogrammetric Engineering & Remote Sensing*, vol. 58, no. 3, pp. 305–310, 1992.
- [16] C. S. Fraser, "Innovations in automation for vision metrology systems," *Photogrammetric Record*, vol. 15, no. 90, pp. 901–911, 1997.
- [17] C. S. Fraser, A. Woods, and D. Brizzi, "Hyper redundancy for accuracy enhancement in automated close range photogrammetry," *Photogrammetric Record*, vol. 20, no. 111, pp. 205–217, 2005.
- [18] E. W. Grafarend, "Optimization of geodetic networks," *Bulletino di Geodesia e Scienze Affini*, vol. 33, no. 4, pp. 351–406, 1974.
- [19] S. O. Mason, *Expert system-based design of photogrammetric networks*, Ph.D. thesis, Institut für Geodäsie und Photogrammetrie, ETH Zürich. Mitteilungen Nr. 53, Zürich, Switzerland, 1994.
- [20] S. O. Mason and A. Grün, "Automatic sensor placement for accurate dimensional inspection," *Computer Vision and Image Understanding*, vol. 61, no. 3, pp. 454–467, 1995.

- [21] S. O. Mason, "Heuristic reasoning strategy for automated sensor placement," *Photogrammetric Engineering & Remote Sensing*, vol. 63, no. 9, pp. 1093–1102, 1997.
- [22] G. Olague, "Automated photogrammetric network design using genetic algorithms," *Photogrammetric Engineering & Remote Sensing*, vol. 68, no. 5, pp. 423–431, 2002, paper awarded the "2003 First Honorable Mention for the Talbert Abrams Award", by ASPRS.
- [23] G. Olague and R. Mohr, "Optimal camera placement for accurate reconstruction," *Pattern Recognition*, vol. 35, no. 4, pp. 927–944, 2002.

Gustavo Olague: Departamento de Ciencias de la Computación División de Física Aplicada, Centro de Investigación Científica y de Educación Superior de Ensenada, Ensenada 22860, BC, Mexico
Email: olague@cicese.mx

Enrique Dunn: Departamento de Electrónica y Telecomunicaciones, División de Física Aplicada, Centro de Investigación Científica y de Educación Superior de Ensenada, Ensenada 22860, BC, Mexico
Email: edunn@cicese.mx

20

Genetic algorithms and neural networks for object detection

Mengjie Zhang

20.1. Introduction

As more and more images are captured in electronic form, the need for programs, which can find objects of interest in a database of images, is increasing. For example, it may be necessary to find all tumours in a database of X-ray images, all cyclones in a database of satellite images, or a particular face in a database of photographs. The common characteristic of such problems can be phrased as “given $subimage_1, subimage_2, \dots, subimage_n$, which are examples of the objects of interest, find all images which contain this object and the locations of all of the objects of interest.” Examples of this kind include target detection problem [28, 67], where the task is to find all tanks, trucks, or helicopters in an image. Unlike most of the current work in the object recognition area, where the task is to detect only objects of a single class [28, 54], the aim of the work presented in this chapter is to detect multiple objects of a number of different classes in a database of large images.

The object recognition task, using traditional image processing and computer vision methods [24, 77], usually involves the following stages: *preprocessing*, *segmentation*, *feature extraction*, and *classification*. The main goal of preprocessing is to remove noise or enhance edges. Segmentation aims to divide an image into coherent regions. Feature extraction is concerned with finding transformations to map patterns to lower-dimensional spaces for pattern representation and to enhance class separability. The output of feature extraction, which is often a vector of feature values, is then passed to the classification step. Using these vectors, the classifier determines the distinguishing features of each object class such that new vectors are placed in the correct class. To obtain good performance, a number of “important” specific features need to be manually determined (selected and extracted) and the classifier has to be chosen for the specific domain. If some objects are lost in one of the early stages, it is very difficult or impossible to recover them in the later stage. In contrast, this chapter focuses on the development of a domain independent method without preprocessing and segmentation for multiple class object detection in a single pass.

A print edition of this book can be purchased at

<http://www.hindawi.com/spc.8.html>

<http://www.amazon.com/dp/9774540018>

In recent years, neural networks and genetic algorithms have attracted attention as very promising methods of solving automatic target recognition and detection problems [13, 31, 56, 64]. A wide variety of problem domains have been shown to be amenable to being treated with these learning and adaptive techniques due to the enormous flexibility of the representation afforded. In terms of input patterns for the neural networks for object detection and recognition, two main approaches have previously been used—feature based and pixel based. In feature-based approach, various features such as brightness, colour, size, and perimeter are extracted from the sub-images of the objects of interest and used as inputs [15, 52]. These features are usually different and specific for different problem domains. In pixel-based approach [56, 33], the pixel values are used directly as inputs. To avoid the disadvantages of hand-crafting feature extraction programs, the approach described in this chapter uses raw pixel data.

The overall goal of this paper is to determine whether domain independent object detection systems, which use raw pixel data and neural and genetic learning, can be built. We require that the systems find objects of different types in one pass. Furthermore, we would like to characterise the kinds of problems for which such systems are likely to be successful. In particular, we will examine the following approaches.

- (i) Train a feedforward neural network on cutouts of the objects of interest and use the trained network as template for sweeping the large images to detect the objects of interest. We refer to this as the basic approach [BP-train].
- (ii) Use a two-phase process in which the first phase is to train the network on the cutouts, using the backpropagation algorithm as in the *BP-train*, and the second phase is to refine the trained network with a genetic algorithm that uses a fitness function based on detection rate and false alarm rate [BP-train + GA-refine].

The remainder of this paper gives a brief literature survey, then describes the three image databases used here. After describing the basic *BP-train* approach for object detection, we describe how to use a genetic algorithm *GA-refine* to refine the networks trained by *BP-train*. Finally, we analyse the results and draw our conclusions.

20.2. Background

20.2.1. Object detection versus object classification

The term *object classification* here refers to the task of discriminating between images of different kinds of objects. Each image contains only one of the objects of interest. The term *object detection* here refers to the detection of small objects in large images. This includes both object classification, as described above, and *object localisation*, which gives the positions of all objects of interest in the large images.

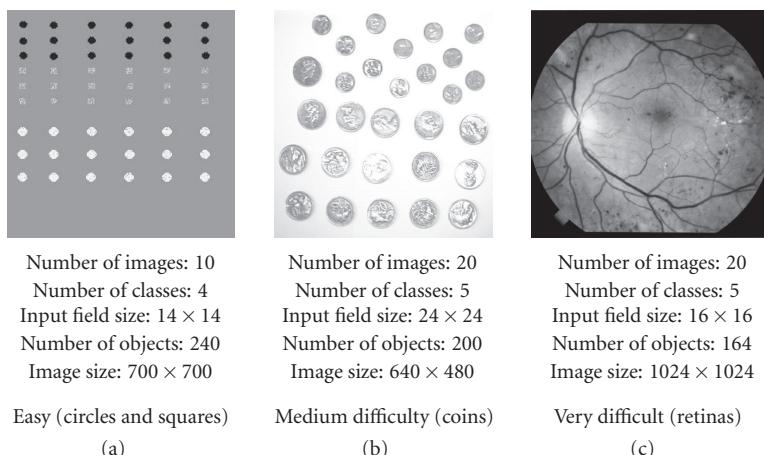


FIGURE 20.1. Object detection problems of increasing difficulty.

20.2.2. Multiclass object detection

In automatic object detection systems, in most cases, all the objects of interest are considered as one class of interest [56, 45]. These systems generally focus on finding the locations of the objects of interest, and all other locations are considered *nonobject* or the *background*. In contrast, the *multiple class* detection problem refers to the case where there is more than one class of objects and both their classes and locations must be determined. In general, multiclass object detection problems are much harder than single class detection problems, and multiclass detection using a single trained program, such as a neural network, is an even more difficult problem.

20.2.3. Performance evaluation

Performance in object detection is measured by detection rate (*DR*) and false alarm rate (*FAR*). The detection rate is the number of objects correctly reported as a percentage of the total number of real objects, and false alarm rate is the number of objects incorrectly reported as a percentage of the total number of real objects. For example, there are 18 grey squares in Figure 20.1(a). A detection system looking for grey squares may report that there are 25. If 9 of these are correct, the detection rate will be $(9/18) * 100 = 50\%$. The false alarm rate will be $(16/18) * 100 = 88.9\%$.

It is important to note that detecting objects in images with very cluttered backgrounds is an extremely difficult problem and that false detection rates of 200–2000% are common [54, 56]. Also note that most work which has been done in this area so far only presents the results of the classification stage and assumes that all other stages have been properly done. However, the results presented in this

chapter are the performance for the whole detection problem (both localisation and classification) in a single pass.

20.2.4. Neural networks related work to object classification/detection

Since the late 1980s, the use of neural networks in object classification and detection has been investigated in a variety of application domains. These domains include military applications, human face recognition, agricultural product classification, handwritten character recognition, and medical image analysis. The types of the neural networks used include multilayer feedforward networks [55], self organising maps [37], higher order networks [29], and ART networks [14].

A summary of neural networks for object classification is given in Table 20.1 according to the kinds of the networks, application domains or problems, and the first authors. Most of these systems are feature-based. In that specific image, features are used as inputs to neural networks.

A summary of neural networks for object detection is presented in Table 20.2. Typically, they belong to the *one-class object detection* problems, where the objects in a single class in large images need to be detected. Very few work in *multiple class object detection* based on a single network or in one stage has not been reported so far.

In this chapter, we will investigate pixel-based neural network approaches for multiclass object detection problems.

20.2.5. Related work of genetic algorithms for evolving neural networks

In addition to the backpropagation algorithm, genetic algorithms can be also used to train (evolve) neural networks. Related work in this area can be grouped into three approaches.

Evolving weights in fixed networks [38, 69]. In this approach, the neural network architecture and learning parameters are predefined. The genetic algorithm is used to train the given network by evolving the weights and biases. In the genetic algorithm, the network weights and biases are encoded into a chromosome. Each chromosome is an individual member of a population, and a population often has several hundred chromosomes. During the evolutionary process, the genetic operators of selection, crossover, and mutation are applied to these individuals. After each generation of the process, the chromosomes are applied to the network, that is, the weights and biases are set from the values which the chromosomes represent. Some accuracy or error measures on the training patterns are then computed and used as the fitness for the genetic algorithm.

Evolving network architectures [49, 76]. In this approach, genetic algorithms are used to evolve not only the network weights and biases, but also the network architecture or topology. References [74, 75] present overviews and classifications of the research in the area of evolutionary design of neural network architectures.

Evolving other factors in neural networks. This includes evolving learning parameters [9], learning rules [26], delta values [39], and input features [12].

TABLE 20.1. Neural networks for object classification.

Kind of network	Applications/problems	Authors	Source
Feedforward networks	Classification of mammograms	Verma	[66]
	target recognition with sonar	Barshan et al.	[8]
		Ayrulu and Barshan	[5]
	Classification of missiles, planes, and helicopters	Howard et al.	[32]
	Maneuver target recognition	Wong and Sundareshan	[73]
	Classification of tanks and trucks on laser radar images	Troxel et al.	[62]
	Underwater target classification	Azimi-Sadjadi et al.	[6]
	Mine and mine-like target detection	Miao et al.	[48]
	Handwritten character recognition	Verma	[65]
	Agricultural product recognition	Winter et al.	[70, 71]
Shared weight neural networks	Multispectral remote sensing data classification	Lee and Landgrebe	[43]
	Natural object classification	Singh et al.	[57]
	Helicopter classification	Stahl and Schoppmann	[60]
	Handwritten optical character recognition	Soulie et al.	[58]
	Zip code recognition	LeCun et al.	[40, 42]
	Digit recognition	de Ridder et al.	[18, 19]
	Lung nodule detection microcalcification classification	Lo et al.	[47]
	Face recognition	Abdi	[1, 63]
		Valentin et al.	[64]
	Vehicle recognition	Bernardon and Carrick	[10]
ART networks	Tank recognition	Fogler et al.	[51]
	Handwritten word recognition	Wessels and Omlin	[68]
Sarf-organising maps	Cloud classification	Dehghan et al.	[20]
	Radar target detection	Tian et al.	[61]
Probability neural networks	3D hand gesture recognition	Kim and Arozullah	[35, 36]
	Bend point and end point recognition	Ahmad and Tresp	[3]
Neocognitron networks	2D and 3D helicopter (F18) recognition	Fukushima et al.	[27]
	Apple sorting (classification)	Spirkovska and Reid	[59]
High-order neural networks	Recognition of bars, triangles, and squares	Hecht-Nielsen	[31]
	River identification (classification)	Cross and Wilson	[17]
	Classification of hot and cold objects	Liu et al.	[46]
	Handwritten character recognition	Casasent and Neiberg	[15]
Hybrid/multiple neural networks	Review	LeCun et al.	[41]
		Egmont-Petersen et al.	[22]

TABLE 20.2. Object detection based on neural networks.

Kind of Network	Applications/problems	Authors	Source
Feedforward neural networks	Target detection in thermal infrared images	Shirvaikar and Trivedi	[56]
Shared weight neural networks	Vehicle detection	Khabou et al. Won et al.	[34] [72]
Probabilistic decision-based neural networks	Face detection and recognition	Lin	[44]
Other hybrid/ multiple neural networks	Aircraft detection and recognition	Waxman et al.	[67]
	Vehicle detection	Bosch et al.	[11]
	Face detection	Féraud et al. El-Bakry et al.	[25] [23]
	Triangle detection	Ahmad and Omohundro	[2]
	Detection/extraction of weak targets for radars	Roth	[53]
	Review of target detection	Rogers et al.	[50]
	Review of target recognition	Roth	[54]

The major advantage of these approaches is that the local minima problem, in which gradient descent algorithms often result, can be avoided. The networks evolved by genetic algorithms presented in the literature are relatively small and the process runs quite well. For the object detection problems investigated in this paper, however, the networks are very large due to the use of pixels as inputs. We will investigate whether genetic algorithms can be used to refine large networks for multiclass object detection problems.

20.3. Image data sets

One of our goals is to characterise the kinds of image detection problems for which our techniques are likely to be successful. To this end, we have investigated three object detection problems of increasing difficulty. Example images and key characteristics are given in Figure 20.1.

The first data set consists of several synthetic images (the easy images), which were generated to give well-defined objects against a uniform background. The pixels of the objects were generated by using a Gaussian generator based on the normal distribution [4] with different means and variances for different classes. All the objects in each class have the same size but are located at different positions.

There are three classes of objects of interest in this database: black circles (*class1*), grey squares (*class2*), and white circles (*class3*) against a uniform grey background (*class other*). The three kinds of objects were generated with different intensities. 10 and 5, 180 and 25, 230 and 20, and 140 and 0 were taken as the mean and standard deviation for *class1*, *class2*, *class3*, and *other*, respectively.

The second data set (the coin images) was intended to be somewhat harder than the easy images and were taken with a CCD camera over a number of days with relatively similar illumination. In these images, the background varies slightly in different areas of the image and between the images. The brightness of objects also varies in a similar way. The objects to be detected are more complex than those in the easy images, but still regular. All the objects in each class have a similar size. They are located at arbitrary positions and with different rotations.

Each of the images contains four object classes of interest, that is, the head side of 5 cent coins (class *head005*), the head side of 20 cent coins (class *head020*), the tail side of 5 cent coins (class *tail005*), and the tail side of 20 cent coins (class *tail020*). The background (class *other*) is relatively uniform, not totally uniform, because of the different lighting conditions and camera positions.

The retina images (data set 3) were taken by a professional photographer with a special apparatus at a clinic. Compared with the previous two data sets and other data set used in automatic target detection problems such as the recognition of regular manmade small objects, the detection problems in this database are very difficult. The images contain very irregular and complex objects of varying sizes, in several classes against a highly cluttered background.

There are two object classes of interest: haemorrhages (class *haem*) and microaneurisms (class *micro*). To give a clearer view of representative samples of the target objects in the retina images, one sample piece of these images is presented in Figure 20.2. In this figure, haemorrhage and microaneurism examples are labeled using white surrounding squares.

These objects are not only located in different places, but the sizes of the objects in each class are different as well, particularly for the haemorrhages. In addition, there are also other objects of different classes such as veins (class *vein*) with different shapes and the retina “edges” (class *edge*). The backgrounds (class *other*) are varied: some parts are quite black, some parts are very bright, and some parts are highly cluttered.

20.3.1. Training and testing subsets

To avoid confusion, we define a number of terms related to the image data. A set of entire images in a database constitutes an *image data set* for a particular problem domain. In this paper, it is randomly split into two parts: a detection training set, which is used to learn a detector, and a *detection test set*, which is used for measuring object detection performance. *Cutouts* refer to subimages which are cut out from a detection training set. Some of these subimages contain examples of the objects of interest and some contain background. These cutouts form a *classification data set*, which is randomly split into two parts: a *classification training set* used for network training, and a *classification test set* for network testing in object classification. The classification test set is also used to tune the trained network for object detection to avoid overfitting. An *input field* refers to a square in the large images. This is used as a moving window for the network sweeping (detection)

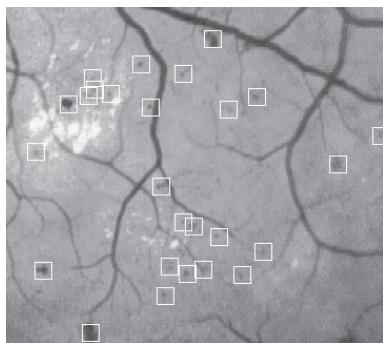


FIGURE 20.2. An enlarged view of one piece of the retina images.

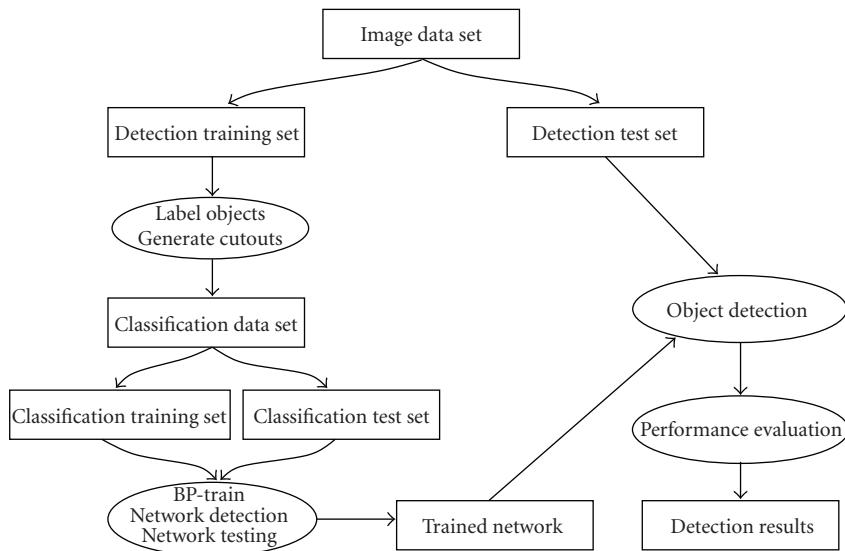


FIGURE 20.3. An overview of the basic approach (BP-train).

process. The size of the input field is the same as that of the cutouts for network training.

20.4. Basic approach (BP-train)

An overview of the basic approach (BP-train) is presented in Figure 20.3. It consists of the following main steps.

- (1) Assemble a data set of images in which the locations and classes of all the objects of interest are manually determined. Divide these entire images into two sets: a detection training set and a detection test set.

- (2) Determine an appropriate size (n) of a square which will cover all single objects of interest and form the input field of the network.
- (3) Manually label the target objects (their centres and classes). Generate a classification data set by cutting out squares of size $n \times n$ from the detection training set. Include examples which are centred on the objects of interest and examples of background.
- (4) If rotation invariance is required, generate new rotated examples from the cutouts. Randomly split the cutouts into a classification training set and a classification test set.
- (5) Determine the network architecture. A three layer feed forward neural network is used in this approach. The $n \times n$ pixel values form the inputs of a training pattern and the classification is the output. The number of hidden nodes is empirically determined.
- (6) Train the network by the backward error propagation algorithm on the classification training data. Test on the classification test set to measure the object classification performance.
- (7) Use the trained network as a moving window template [7] to detect the objects of interest in the detection test set. If the output of the network for a class exceeds the threshold, then report an object of that type at the current location.
- (8) Using search, find a threshold which results in all objects in the detection test set, being found with the smallest false alarm rate.
- (9) Evaluate the object detection performance of the network by comparing the classes and locations detected with the known classes and locations in the detection test set and calculating the detection rate and the false alarm rate.

20.4.1. Object classification—network training and testing

We use the backward error propagation algorithm [55] with online learning and the fan-in factor [40, 18] to train the networks. In online learning (also called the stochastic gradient procedure), weight changes are applied to the network after each training pattern. The fan-in is the number of elements that either excites or inhibits a given node of the network. The weights are divided by the number of inputs of the node to which the connection belongs before network training, and the size of the weight change of a node is updated in a similar way.

The training is terminated when the classification accuracy in the classification training set reaches a predefined percentage. When training is terminated, the trained network weights and biases are saved for the use in network testing or subsequent resumption of training.

The trained network is then applied to the classification test set. If the test performance is reasonable, then the trained network is ready to be used for object detection. Otherwise, the network architecture and/or the learning parameters need to be changed and the network retrained, either from the beginning or from a previously saved, partially trained network. During network training and testing, the

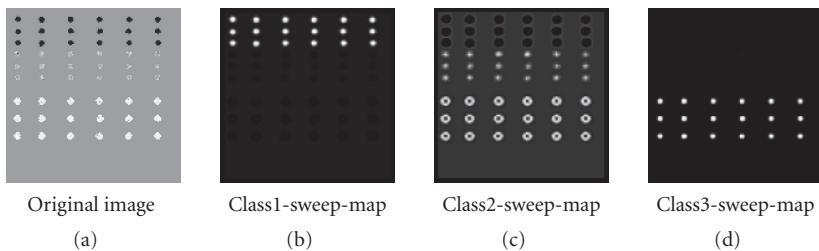


FIGURE 20.4. Sample object sweeping maps in object detection.

classification is regarded as correct if the output node with the largest activation value corresponds to the desired class of a pattern.

20.4.2. Object detection

After network training is successfully done, the trained network is used to detect the classes and locations of the objects of interest in the detection test set, which is not used in any way for network training. Classification and localisation are performed by the procedures: *network sweeping*, *finding object centres*, and *object matching*.

20.4.2.1. Network sweeping

During network sweeping, the trained neural network is used as a template matcher and is applied, in a moving window fashion, over the large images to detect the objects of interest. The template is swept across and down these large images, pixel by pixel in every possible location.

After the sweeping is finished, an *object sweeping map* for each object class of interest will be produced. An object sweeping map contains the outputs of the network for each pixel position in the large image and can be visualised as a grey level image. Sample object sweeping maps for *class1*, *class2*, and *class3* together with the original image for the easy detection problem are shown in Figure 20.4.

During the sweeping process, if there is no match between a square in a detecting image and the template, then the neural network output is 0, which corresponds to black in the sweeping maps; partial match corresponds to grey on the centre of the object, and best match is close to white. The object sweeping map can be used to get a qualitative indication of how accurate the detection step is likely to be. Figure 20.4 reveals that *class1* and *class3* objects will be detected very accurately but there will probably be errors in *class2*.

20.4.2.2. Finding object centres

Since we use the cutouts centred on the target objects as training examples, we expect that the trained network will give the largest activation output value for the

For each object sweeping map:

- (1) set a *threshold* for the class;
- (2) set all of the values in the sweeping map to zero if they are less than the threshold;
- (3) search for the largest value, save the corresponding position (x, y) , and label this position as an object centre;
- (4) set all values in the square input field of the labeled centre (x, y) to zero;
- (5) repeat steps (3) and (4) until all values in the object sweeping map are zero.

ALGORITHM 20.1

target class of an object to be detected when the sweeping window associated with the trained network is centred on the object. Otherwise, the trained network with the sweeping window would result in false alarms or missing objects.

Based on this expectation, we developed a *centre-finding algorithm* to find the centres of all objects detected by the trained network. For each class of interest, this algorithm is used to find the centres of the objects based on the corresponding object sweeping map. The centre-finding algorithm is presented in Algorithm 20.1.

20.4.2.3. Choice of thresholds

During the object detection process, various thresholds result in different detection results. The higher the threshold, the fewer the objects that can be detected by the trained network, which results in a lower-detection rate, but also a lower false alarm rate. Similarly, the lower the threshold selected, the higher the detection rate and also the higher the false alarm rate. Thus there is a trade-off between the detection rate and the corresponding false alarm rate. Ideally, there will be a threshold, will give 100% detection with 0% false alarms. If such a threshold cannot be found, we use the one that gives the highest detection rate with fewest false alarms as the best detection result. The thresholds were found by an exhaustive search.

20.4.2.4. Object matching

Object matching compares all the object centres reported by the centre-finding algorithm with all the desired known object centres and reports the number of objects correctly detected. Here, we allow location error of *tolerance* pixels in the x and y directions. Clearly, the use of a large value for *tolerance* will result in better detection rate and false alarm rate, but this will lead to the fact that the detected object centres are allowed very far from the true centres of the target objects. Accordingly, to keep a balance between the two aspects, we choose a value of 4 for *tolerance* in this approach. For example, if the coordinates of a known object are 21, 19 and the coordinates of a detected object are 22, 21, we consider that the object has been correctly located.

TABLE 20.3. Classification results of network training on cutouts, 15 runs.

	Easy	Coins	Retinas
Size of input field	14	24	16
Training set size	60	100	100
Test set size	180	100	61
No. of input nodes	196	576	256
No. of output nodes	4	5	5
No. of hidden nodes	4	3	4
Learning rate	0.5	0.5	1.5
Momentum	0	0	0
Training epochs	199	234	475
Training accuracy (%)	100	100	75
Test accuracy (%)	100	100	71

20.4.3. Results

We first give the results for the network training on the cutouts and then for the object detection performance.

20.4.3.1. Object classification results

The results for network training on the cutouts are shown in Table 20.3. From the table, it can be seen that there is a large variation in the number of epochs of training needed and that the number of epochs increases with increasing complexity of the objects, as would be expected. The number of hidden nodes was determined empirically by finding the smallest number which gave successful training, however, the performance was quite robust for up to 10 hidden nodes. Also, the easy and coin objects can be classified without error but not so for the retina objects. For the easy and coin objects, training was terminated when all of the training examples were correctly classified. For the retina objects, this could not be achieved and training was terminated when 75% of the training examples were correctly classified.

20.4.3.2. Object detection results

This section describes the detection performance of the basic approach on the three detection problems. The 15 networks from the runs described in the previous section were used in the object detection step of the methodology. The results are shown in Table 20.4. These are averages over 15 training and detection runs.

On the easy images, the basic approach always achieved a 100% detection rate and a corresponding zero false alarm rate for *class1* (black circles) and *class3* (white circles). However, this could not be achieved for *class2* (grey squares). Even if a neural network achieved a 100% detection rate under a certain threshold, 91.2% false alarm rate was also produced.

TABLE 20.4. Average detection results (DR and FAR) on detection test set over 15 runs.

	Easy	Coins	Retinas
Size of input field	14	24	16
Network architecture	196-4-4	576-3-5	256-5-4
DR for class1 (%)	(black circles) 100	(head005) 100	(<i>haem</i>) 74
FAR for class1 (%)	0	0	2859
DR for class2 (%)	(grey squares) 100	(tail005) 100	(<i>micro</i>) 100
FAR for class2 (%)	91.2	0	10104
DR for class3 (%)	(white circles) 100	(head020) 100	—
FAR for class3 (%)	0	182	—
DR for class4 (%)	—	(tail020) 100	—
FAR for class4 (%)	—	37.5	—

On the coin images, in each run, it was always possible to find a threshold for the network output for classes *head005* and *tail005*, which resulted in detecting all of the objects of these classes with no false alarms. However, detecting classes *head020* and *tail020* was a relatively difficult problem with this method. Although all the objects in these two classes were correctly detected (100% detection rate), the neural networks produced some false alarms. The average false alarm rates for the two classes at a 100% detection rate were 182% and 37.5%, respectively.

Compared with the performance of the easy and coin images, the results on the very difficult retina images are disappointing. The best detection rate for class *haem* was 73.91% with a corresponding false alarm rate of 2859%. Even at a detection rate of 50%, the false alarm rate was still quite high (about 1800%). All the objects of class *micro* were correctly detected (a detection rate of 100%) but with a false alarm rate of 10104%. By adjusting the thresholds, the false alarm rate could be reduced, but only at the cost of a decrease in the detection rate.

20.4.4. Discussion

The experimental results showed that this approach performed very well for detecting a number of simple and regular objects against a relatively uniform background such as detecting black circles and white circles in the easy images. It performed poorly on the detection of class *haem* and class *micro* objects in the retina images. As expected, the performance degrades when the approach is applied to detection problems of increasing difficulty.

In the basic approach, the network is trained on the cutouts of the objects and the trained network was directly applied to the entire images in the detection test set. The remainder of this chapter will use a genetic algorithm to refine the trained networks over full images in the detection training set, then apply the refined networks on full images in the detection test set to measure object detection performance. We expect that the genetic algorithm can refine the trained networks which improve the false alarm rates at the same best detection rate as in the basic approach.

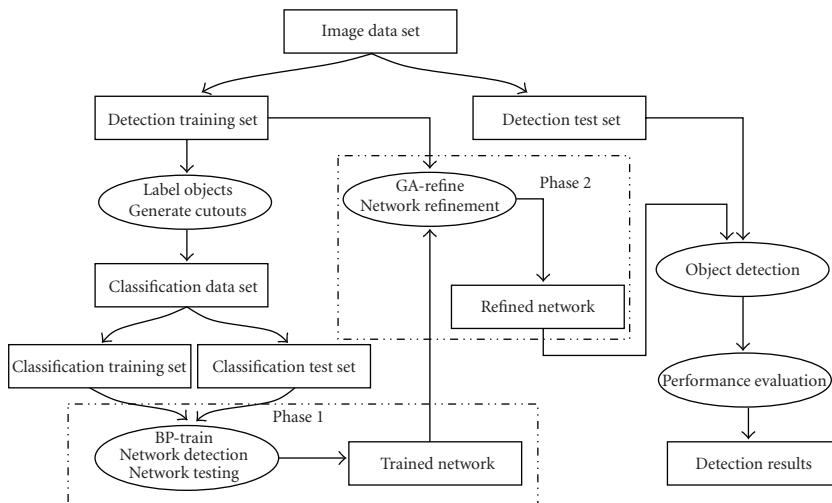


FIGURE 20.5. An overview of the two-phase approach.

20.5. Genetic algorithms for network refinement

This section introduces a genetic algorithm to refine the trained networks for object detection [78, 79]. The network training step in the basic approach was combined with the genetic algorithm to form a two-phase approach. An overview of the two-phase approach is shown in Figure 20.5.

In the first phase, the network is trained on the cutouts for object classification, using the backward error propagation algorithm to maximise the classification accuracy on the cutouts in the classification training set. In the second phase, the weights of the trained network are refined using a genetic algorithm, which uses a fitness function which maximises detection performance on the entire images in the detection training set.

This two-phase approach results in a new object detection method: network training by the backward error propagation algorithm in the basic approach followed by network refinement by the refined genetic algorithm (BP-train + GA-refine). BP-train has been described in the previous section and the rest of this section describes the GA-refine algorithm.

As discussed in Section 20.2.5, there are many difference approaches to the use of genetic algorithms for training neural networks. In this approach, as the network has been trained by the backpropagation algorithm as described in the previous section, the network architecture has already fixed. The goal of the *GA-refine* algorithm is to refine the weights in the trained network in order to reduce the false alarm rate in object detection, and the genetic algorithms for learning network architectures will not be considered here. On the other hand, the *2DELTA-GANN* algorithm [39, 16] has been proved very successful in learning weights for very small feedforward neural networks such as the *xor* and the *424 encoder*.

```

if (x1 == 1) then
    if (x2 == 1 and x3 == 1) then
        delta2 = delta2 * 2;
    else
        delta2 = delta2/2;
    endif
    delta1 = delta1 + delta2;
endif
weight = weight + delta1;
end

```

ALGORITHM 20.2. Weight-updating mechanism in the genetic algorithm.

networks. In this chapter, we use this genetic algorithm as our **GA-refine** algorithm to investigate whether it can be applied to our large networks and lead to better object detection performance. Unless otherwise specified, a bias in the network is treated as just another weight.

20.5.1. Gene structure and weight update algorithm

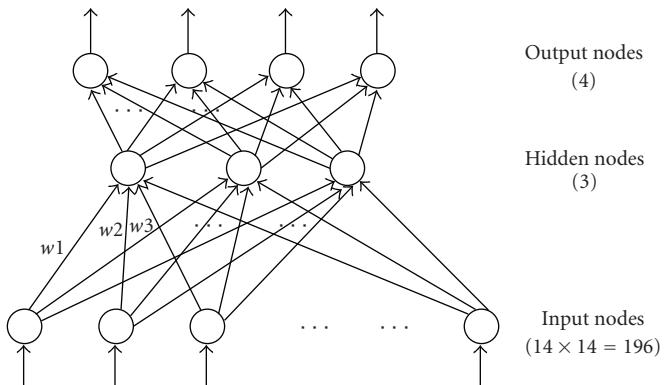
Rather than encoding all the weights of a neural network themselves into the chromosomes directly, this approach only encodes the prospective change (*deltas*) of a weight into a gene and uses three rule bits in the gene for the use of genetic operators and for updating these weight changes. A gene in the *GA-train* algorithm corresponds to a single weight in the network (but not the weight itself) and is a composite structure.

- (i) There are three rule bits called x_1 , x_2 , and x_3 .
- (ii) There are two floating-point values called *delta1* and *delta2*.

With this gene structure, the number of rule bits in each chromosome, which corresponds to a single network, will be three times the number of weights in the network. The number of the floating point values, which represents the *deltas* associated with each chromosome, is twice the number of the weights. Note that the weights of the network are not a part of the chromosome, whose genes represent a method of changing the weights of the network in the way specified by the weight-updating rule.

As shown in Algorithm 20.2, the x_1 , x_2 , and x_3 values specify a heuristic rule to apply to *delta2*. *Delta1* will then be modified by *delta2*, and this will in turn be applied to the weight associated with the gene. The interpretation of the rule bits and the mechanism of updating weights is presented in Algorithm 20.2. For example, if x_1 , x_2 , and x_3 are all 1, then *delta2* is doubled, the new value of *delta2* is then added to *delta1*, and the new weight is calculated by adding the new value of *delta1* to the original weight.

The changing mechanism allows for a weight to be changed by a *delta1* value, which in turn is modified by the *delta2* value. The rate of change of the weight and *delta1* (the gradient) is changeable due to the *delta2* value. This is used to provide a heuristic “second order” changing mechanism to the weight modification rule.



Weights	w1	w2	...	w600	...
Genes	$x_1 \ x_2 \ x_3 \ \text{delta1} \ \text{delta2}$	$x_1 \ x_2 \ x_3 \ \text{delta1} \ \text{delta2}$...	$x_1 \ x_2 \ x_3 \ \text{delta1} \ \text{delta2}$...
Network1	1 1 0 0.13 0.02	1 1 1 0.35 0.05	...	1 0 1 -0.48 0.21	...
Network2	0 0 1 0.42 -0.12	1 0 0 -0.25 0.06	...	0 1 0 0.27 -0.10	...

FIGURE 20.6. Sample genes and chromosomes associated with a network for the simple object detection problem in the easy image in the genetic algorithm.

20.5.2. Sample networks with chromosomes

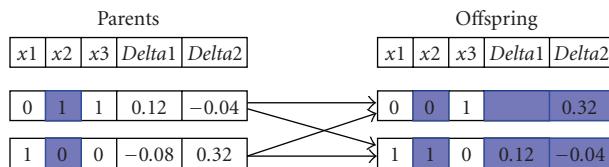
Figure 20.6 shows a network architecture of 196-3-4 for the easy images with sample chromosomes and genes for weight changes. In this figure, there are two chromosomes corresponding to two evolved networks. The weights are labeled as w_1, w_2, \dots . As mentioned earlier, the network architecture and the actual weights are not encoded in the genetic algorithm but saved outside. Supposing the original values of w_1 and w_2 are 0.34 and -0.49, we explain how the mechanism of updating weights is applied and the new weights calculated.

For w_1 in *network1*, $x_1 = 1, x_2 = 1, x_3 = 0, \text{delta1} = 0.13$ and $\text{delta2} = 0.02$. According to the weight-changing mechanism, the value of the delta2 should be halved and the new value of delta2 becomes $0.02/2 = 0.01$. Then, this value is added to delta1 and the new value of delta1 becomes $0.13 + 0.01 = 0.14$. The new value of w_1 is accordingly updated to $0.34 + 0.14 = 0.48$. Similarly, since $x_1 = x_2 = x_3 = 1$ in the gene of w_2 , the new values for $\text{delta2}, \text{delta1}$, and the actual weight for w_2 are $\text{delta2} = \text{delta2} * 2 = 0.05 * 2 = 0.10, \text{delta1} = \text{delta1} + \text{delta2} = 0.35 + 0.10 = 0.45$, and $w_2 = w_2 + \text{delta1} = (-0.49) + 0.45 = -0.04$.

For w_1 in *network2*, since $x_1 = 0, \text{delta1}$ and delta2 will remain unchanged, and the new value of w_1 becomes $w_1 + \text{delta1} = 0.34 + 0.42 = 0.76$.

20.5.3. Genetic operators

The commonly used biased roulette wheel mechanism [30] is used for parent selection. The crossover operator is based on parameterised uniform crossover [21].

FIGURE 20.7. Crossover in the *GA-train* algorithm.FIGURE 20.8. Mutation in the *GA-train* algorithm.

Crossover is applied to both the *rule* bits of the chromosome and the *deltas*. If there is an exchange of any of the rule bits during crossover, the *delta1* and *delta2* values are exchanged between the parent chromosomes. Notice that this crossover requires the two real values for the *deltas* exchanged completely/simultaneously when the rule bit part is changed, which is different from that in the real-valued genetic algorithm. This is mainly because the three bits and the two real values form a single gene in a chromosome, representing the change of a single weight in a neural network. In addition, the two real-valued *deltas* in each gene are dependent of each other, which is shown in the weight updating rule. Figure 20.7 shows an example of the crossover operator in the genetic algorithm.

The mutation operator is based on the standard single bit mutation. If any bit on a gene is mutated, the *delta1* and *delta2* values for that gene will be replaced with randomly generated values in order to maintain a sufficiently diverse population of chromosomes. Figure 20.8 shows an example of the mutation operator in the genetic algorithm.

20.5.4. Fitness function

In this genetic algorithm, a chromosome corresponds to a neural network. The weights of the network are initialised with the trained weights obtained in the first phase instead of being randomly generated.

In network refinement process, the fitness of a chromosome is calculated as follows.

- (1) Realise the network from the weights and the weight changes encoded in the chromosome.
- (2) Apply the network as a moving $n \times n$ window template across the entire images in the detection training set to locate the object of interest. This is identical to the corresponding detection procedure used in the basic approach except that this is now performed as part of the training procedure.

- (3) Compare the detected object centres with known locations of the desired objects, obtain the number of objects correctly and incorrectly detected by the network, and determine the overall detection rate and false alarm rate of this network.
- (4) Compute the fitness according to

$$\text{fitness (FAR, DR)} = \text{FAR}/(\text{FAR} + \text{DR}) + W_d * (1 - \text{DR}), \quad (20.1)$$

where DR and FAR represent the detection rate and the false alarm rate of the network, and W_d is a constant which reflects the relative importance of the false alarm rate and the detection rate. When W_d is close to zero, the influence of false alarm rate increases; when the W_d is getting bigger, the influence of detection rate will become larger.

Under this design, the smaller the fitness, the better the detection performance. The best case is zero fitness when the detection rates for all classes are 100% and the false alarm rates are zero, that is, all the objects of interest are correctly detected by the network without any false alarms.

20.5.5. Results

The GA parameters and the detection results at the best detection rate were achieved for BP-train + GA-refine in Table 20.5. The parameter values were carefully selected through empirical search via trial experiments. The evolutionary process for network refinement was terminated when either the problem was solved or the number of generations reached 50.

It is evident from Tables 20.4 and 20.5 that the refinement genetic algorithm GA-refine significantly improved the object detection performance on all the three data sets. For the easy and coin image data sets, the two-phase approach BP-train + GA-refine successfully detected all the objects of interest in all classes without any false alarms. For the very difficult retina image data set, the best detection and false alarm rates were also greatly improved for both the *haem* and *micro* classes, although an ideal performance was not achieved.

To give a clearer view of the comparison of the two-phase method over the basic approach, we present the extended ROC curves of detecting the heads and tails of 20 cent coins and detecting the haemorrhages and microaneurisms in the retina images in Figure 20.9.

As can be seen from Figure 20.9, at all levels of detection rate, the BP-train + GA-refine always achieved a smaller false alarm rate than the BP-train method. In particular, the BP-train + GA-refine method did not produce any false alarms for classes *head020* and *tail020* at all levels of detection rate (achieved perfect performance). For class *haem*, the two-phase method not only improved the false alarm rates at all levels of detection rates over the basic method, but also achieved better detection rate. For class *micro*, the new two-phase method significantly reduced the false alarm rates at different detection rates. These experimental results show the refinement genetic algorithm always resulted in better detection

TABLE 20.5. GA parameter values and the average detection results (DR and FAR) using *BP-train + GA-refine* on detection test set over 15 runs.

	Easy	Coins	Retinas
Size of input field	14	24	16
Network architecture	196-4-4	576-3-5	256-5-4
Population size	200	200	500
Crossover rate	95%	95%	90%
Mutation rate	5%	5%	10%
Max generations	50	50	50
Delta1 range	± 0.08	± 0.04	± 0.02
Delta2 range	± 0.05	± 0.02	± 0.005
W_d	0.4	1.5	0.67
DR for class1 (%)	(black circles) 100	(head005) 100	(<i>haem</i>) 82
FAR for class1 (%)	0	0	2156
DR for class2 (%)	(grey squares) 100	(tail005) 100	(<i>micro</i>) 100
FAR for class2 (%)	0	0	2706
DR for class3 (%)	(white circles) 100	(head020) 100	—
FAR for class3 (%)	0	0	—
DR for class4 (%)	—	(tail020) 100	—
FAR for class4 (%)	—	0	—

performance than the corresponding methods without the refinement in all three data sets.

20.5.6. Discussion

To achieve the goal of domain independence, we used the raw pixel values directly as inputs to neural networks. However, this led to large networks and longer training times. One issue that remains is whether all pixel values are needed or whether *pixel statistics* or pixel level domain-independent image features such as the means and standard deviations of some local regions or edges in the image cutouts could be used to achieve similar accuracy with smaller networks. We have used such kind of features shown in Figure 20.10. An investigation, which used the same algorithms but with some features based on pixel statistics instead of pixel values, found that the false alarm rates were higher while the same detection rates could be achieved.

20.6. Conclusions

The goal of this chapter was to develop and evaluate a domain independent approach to multiple class object detection problems and to characterise the kinds of problems for which it is likely to be successful. The goal was successfully achieved by developing a method based on training a neural network classifier on cutouts

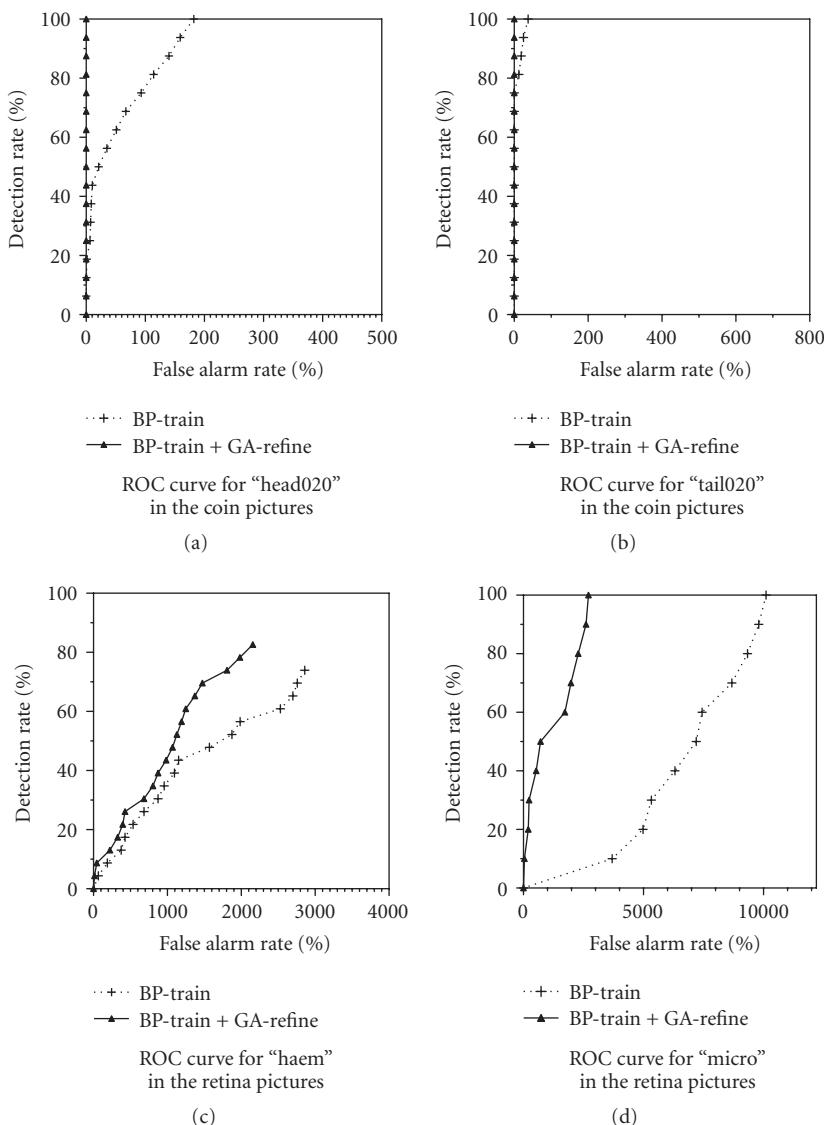
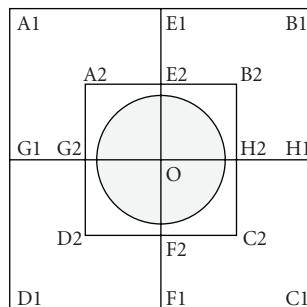


FIGURE 20.9. Extended ROC curves for (a) class *head020*, (b) class *tail020*, (c) class *haem*, and (d) class *micro*.

of the objects of interest and then refining the network weights using a genetic algorithm with fitness based on maximising the detection rate and minimising the false alarm rate. The results show that the technique is likely to be successful if (1) the objects are on a relatively uncluttered background, (2) the objects in each class are roughly the same size, and (3) the objects are regular but can have complex internal details.



Features		Regions and axes of interest
Mean	Sd	
F1	F2	Big square A1-B1-C1-D1
F3	F4	Small central square A2-B2-C2-D2
F5	F6	Upper left square A1-E1-O-G1
F7	F8	Upper right square E1-O-H1-B1
F9	F10	Lower left square G1-O-F1-D1
F11	F12	Lower right square O-H1-C1-F1
F13	F14	Central row of the big square G1-H1
F15	F16	Central column of the big square E1-F1
F17	F18	Central row of the small square G2-H2
F19	F20	Central column of the small square E2-F2

FIGURE 20.10. Features based on local image region intensities (sd: standard deviation).

Domain independence has been achieved by using raw pixel values as inputs to the network and thus avoiding the problems of hand-crafting feature selection programs. Rotation invariance is achieved by having the objects in the training images at random orientations and/or by rotating the cutouts prior to network training. Also, the method finds objects of different classes in a single pass, unlike most current work in this area which use different programs in multiple independent stages to achieve localisation and classification.

There are two negative aspects of the method—there are a number of new parameters whose values must be determined by the user, and the run times for the genetic algorithms can be quite long, although this could be ameliorated by the use of parallel hardware.

The method achieved a 100% detection rate and a 0% false alarm rate for simple object detection against a uniform background in the easy images and medium complexity object detection against a relatively uniform background in the coin images. However, on a very difficult problem involving retinal pathologies on a highly cluttered background, the detection performance was disappointing. However, it was competitive with alternative techniques such as different types of neural networks and some statistical methods on similar problems.

The two-phase method with the backpropagation algorithm for network training on the cutouts and the genetic algorithm for network weight refinement always achieved significantly better performance, suggesting that the genetic algorithm can successfully refine neural networks for object detection problems.

For future work, we will investigate alternative fitness functions for the refinement genetic algorithm and examine this approach on other data sets such as detecting small objects in satellite images.

Acknowledgments

The author would like to thank Professor Victor Ciesielski for his guidance and suggestions, Dr. James Thom and Professor Zhi-Qiang Liu for a number of useful discussions, and Chris Kamusinski for providing and labeling the retina images.

Bibliography

- [1] H. Abdi, "A generalized approach for connectionist auto-associative memories: interpretation, implications and illustration for face processing," in *Artificial Intelligence and Cognitive Sciences*, J. Demongeot, T. Herve, V. Rialle, and C. Roche, Eds., chapter 6, pp. 149–165, Manchester University Press, Manchester, UK, 1988.
- [2] S. Ahmad and S. Omohundro, "A network for extracting the locations of point clusters using selective attention," in *Proceedings of the 12th Annual Conference of the Cognitive Science Society*, MIT, Cambridge, Mass, USA, July 1990, also in Tech. Rep. #90-011.
- [3] S. Ahmad and V. Tresp, "Some solutions to the missing feature problem in vision," in *Advances in Neural Information Processing Systems*, J. D. Cowan, S. J. Hanson, and C. L. Giles, Eds., chapter 5, pp. 393–400, Morgan Kaufmann Publishers, San Mateo, Calif, USA, 1993.
- [4] J. H. Ahrens and U. Dieter, "Extensions of Forsythe's method for random sampling from the normal distribution," *Mathematics of Computation*, vol. 27, no. 124, pp. 927–937, 1973.
- [5] B. Ayrulu and B. Barshan, "Neural networks for improved target differentiation and localization with sonar," *Neural Networks*, vol. 14, no. 3, pp. 355–373, 2001.
- [6] M. R. Azimi-Sadjadi, D. Yao, Q. Huang, and G. J. Dobeck, "Underwater target classification using wavelet packets and neural networks," *IEEE Transactions on Neural Networks*, vol. 11, no. 3, pp. 784–794, 2000.
- [7] D. H. Ballard and C. M. Brown, *Computer Vision*, Prentice-Hall, Englewood Cliffs, NJ, USA, 1982.
- [8] B. Barshan, B. Ayrulu, and S. W. Utete, "Neural network-based target differentiation using sonar for robotics applications," *IEEE Transactions on Robotics and Automation*, vol. 16, no. 4, pp. 435–442, 2000.
- [9] R. K. Belew, J. McInerney, and N. N. Schraudolph, "Evolving networks: using the genetic algorithm with connectionist learning," in *Artificial Life II*, pp. 511–547, Addison-Wesley, Reading, Mass, USA, 1992.
- [10] A. M. Bernardon and J. E. Carrick, "A neural system for automatic target learning and recognition applied to bare and camouflaged SAR targets," *Neural Networks*, vol. 8, no. 7-8, pp. 1103–1108, 1995.
- [11] H. Bosch, R. Milanese, and A. Labbi, "Object segmentation by attention-induced oscillations," in *Proceedings of IEEE International Joint Conference on Neural Networks, IEEE World Congress on Computational Intelligence (IJCNN '98)*, vol. 2, pp. 1167–1171, Anchorage, Alaska, USA, May 1998.
- [12] F. Z. Brill, D. E. Brown, and W. N. Martin, "Fast genetic selection of features for neural network classifiers," *IEEE Transactions on Neural Networks*, vol. 3, no. 2, pp. 324–328, 1992.
- [13] R. Brunelli and T. Poggio, "Face recognition: features versus templates," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 15, no. 10, pp. 1042–1052, 1993.
- [14] G. A. Carpenter and S. Grossberg, "A massively parallel architecture for a self-organising neural pattern recognition machine," *Computer Vision, Graphics, and Image Processing*, vol. 37, no. 1, pp. 54–115, 1987.
- [15] D. P. Casasent and L. M. Neiberg, "Classifier and shift-invariant automatic target recognition neural networks," *Neural Networks*, vol. 8, no. 7-8, pp. 1117–1129, 1995.

- [16] J. Riley and V. B. Ciesielski, "An evolutionary approach to training feed-forward and recurrent neural networks," in *Proceedings of the 2nd International Conference on Knowledge-Based Intelligent Electronic Systems (KES '98)*, L. C. Jain and R. K. Jain, Eds., vol. 3, pp. 596–602, Adelaide, Australia, April 1998.
- [17] N. Cross and R. Wilson, "Neural networks for object recognition," Tech. Rep., Department of Computer Science, University of Warwick, Coventry, RI, USA, October 1995.
- [18] D. de Ridder, "Shared weights neural networks in image analysis," M.S. thesis, Delft University of Technology, Delft, The Netherlands, February 1996.
- [19] D. de Ridder, A. Hoekstra, and R. P. W. Duin, "Feature extraction in shared weights neural networks," in *Proceedings of the 2nd Annual Conference of the Advanced School for Computing and Imaging (ASCI '96)*, pp. 289–294, Delft, The Netherlands, June 1996.
- [20] M. Dehghan, K. Faez, and M. Ahmadi, "Hybrid handwritten word recognition using self-organizing feature map, discrete HMM, and evolutionary programming," in *Proceedings of IEEE-INNS-ENNS International Joint Conference on Neural Networks (IJCNN '00)*, vol. 5, pp. 515–520, Como, Italy, July 2000.
- [21] W. M. Spears and K. A. De Jong, "On the virtues of parameterized uniform crossover," in *Proceedings of the 4th International Conference on Genetic Algorithms (ICGA '91)*, R. K. Belew and L. B. Booker, Eds., pp. 230–236, Morgan Kaufman, San Diego, Calif, USA, July 1991.
- [22] M. Egmont-Petersen, D. de Ridder, and H. Handels, "Image processing with neural networks—a review," *Pattern Recognition*, vol. 35, no. 10, pp. 2279–2301, 2002.
- [23] H. M. El-Bakry, M. A. Abo-Elsoud, and M. S. Kamel, "Fast modular neural nets for human face detection," in *Proceedings of IEEE-INNS-ENNS International Joint Conference on Neural Networks (IJCNN '00)*, vol. 3, pp. 320–324, Como, Italy, July 2000.
- [24] O. Faugeras, *Three-Dimensional Computer Vision: A Geometric Viewpoint*, MIT Press, Cambridge, Mass, USA, 1993.
- [25] R. Féraud, O. J. Bernier, J.-E. Viallet, and M. Collobert, "A fast and accurate face detector based on neural networks," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 23, no. 1, pp. 42–53, 2001.
- [26] J. F. Fontanari and R. Meir, "Evolving a learning algorithm for the binary perceptron," *Network*, vol. 2, no. 4, pp. 353–359, 1991.
- [27] K. Fukushima, E. Kimura, and H. Shouno, "Neocognitron with improved bend-extractors," in *Proceedings of IEEE International Joint Conference on Neural Networks, IEEE World Congress on Computational Intelligence (IJCNN '98)*, vol. 2, pp. 1172–1175, Anchorage, Alaska, USA, May 1998.
- [28] P. D. Gader, J. R. Miramonti, Y. Won, and P. Coffield, "Segmentation free shared weight networks for automatic vehicle detection," *Neural Networks*, vol. 8, no. 9, pp. 1457–1473, 1995.
- [29] C. L. Giles and T. Maxwell, "Learning, invariance, and generalization in high-order neural networks," *Applied Optics*, vol. 26, no. 23, pp. 4972–4978, 1987.
- [30] D. E. Goldberg, *Genetic Algorithms in Search, Optimization and Machine Learning*, Addison Wesley, Reading, Mass, USA, 1989.
- [31] R. Hecht-Nielsen, "Neural networks and image analysis," in *Neural Networks for Vision and Image Processing*, G. A. Carpenter and S. Grossberg, Eds., chapter 17, pp. 449–460, MIT Press, Cambridge, Mass, USA, 1992.
- [32] A. Howard, C. Padgett, and C. C. Liebe, "A multi-stage neural network for automatic target detection," in *Proceedings of IEEE International Joint Conference on Neural Networks, IEEE World Congress on Computational Intelligence (IJCNN '98)*, vol. 1, pp. 231–236, Anchorage, Alaska, USA, May 1998.
- [33] J. S. N. Jean and J. Wang, "Weight smoothing to improve network generalization," *IEEE Transactions on Neural Networks*, vol. 5, no. 5, pp. 752–763, 1994.
- [34] M. A. Khabou, P. D. Gader, and J. M. Keller, "LADAR target detection using morphological shared-weight neural networks," *Machine Vision and Applications*, vol. 11, no. 6, pp. 300–305, 2000.

- [35] M. W. Kim and M. Arozullah, "Generalized probabilistic neural network based classifiers," in *Proceedings of IEEE International Joint Conference on Neural Networks (IJCNN '92)*, vol. 3, pp. 648–653, Baltimore, Md, USA, June 1992.
- [36] M. W. Kim and M. Arozullah, "Neural network based optimum radar target detection in non-Gaussian noise," in *Proceedings of IEEE International Joint Conference on Neural Networks (IJCNN '92)*, vol. 3, pp. 654–659, Baltimore, Md, USA, June 1992.
- [37] T. Kohonen, *Self-Organization and Associative Memory*, Springer, New York, NY, USA, 3rd edition, 1988.
- [38] P. G. Kornig, "Training neural networks by means of genetic algorithms working on very long chromosomes," *International Journal of Neural Systems*, vol. 6, no. 3, pp. 299–316, 1995.
- [39] R. Krishnan and V. Ciesielski, "2DELTA-GANN: a new approach to using genetic algorithms to train neural networks," in *Proceedings of the 5th Australian Neural Networks Conference*, A. C. Tsoi, Ed., pp. 38–41, University of Queensland, Brisbane, Australia, February 1994.
- [40] Y. LeCun, B. Boser, J. S. Denker, et al., "Backpropagation applied to handwritten zip code recognition," *Neural Computation*, vol. 1, no. 4, pp. 541–551, 1989.
- [41] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," in *Intelligent Signal Processing*, pp. 306–351, IEEE Press, Piscataway, NJ, USA, 2001.
- [42] Y. LeCun, L. D. Jackel, B. Boser, et al., "Handwritten digit recognition: applications of neural network chips and automatic learning," *IEEE Communications Magazine*, vol. 27, no. 11, pp. 41–46, 1989.
- [43] C. Lee and D. A. Landgrebe, "Decision boundary feature extraction for neural networks," *IEEE Transactions on Neural Networks*, vol. 8, no. 1, pp. 75–83, 1997.
- [44] S.-H. Lin, "An introduction to face recognition technology," *Informing Science*, vol. 3, no. 1, pp. 1–7, 2000, special issue on *Multimedia Technologies and Informing Science, Part II*.
- [45] S.-H. Lin, S.-Y. Kung, and L.-J. Lin, "Face recognition/detection by probabilistic decision-based neural network," *IEEE Transactions on Neural Networks*, vol. 8, no. 1, pp. 114–132, 1997.
- [46] X. Liu, D. Wang, and J. R. Ramirez, "Extracting hydrographic objects from satellite images using a two-layer neural network," in *Proceedings of IEEE International Joint Conference on Neural Networks, IEEE World Congress on Computational Intelligence (IJCNN '98)*, vol. 2, pp. 897–902, Anchorage, Alaska, USA, May 1998.
- [47] S.-C. B. Lo, H.-P. Chan, J.-S. Lin, H. Li, M. T. Freedman, and S. K. Mun, "Artificial convolution neural network for medical image pattern recognition," *Neural Networks*, vol. 8, no. 7-8, pp. 1201–1214, 1995.
- [48] X. Miao, M. R. Azimi-Sadjadi, B. Tian, A. C. Dubey, and N. H. Witherspoon, "Detection of mines and minelike targets using principal component and neural-network methods," *IEEE Transactions on Neural Networks*, vol. 9, no. 3, pp. 454–463, 1998.
- [49] M. A. Potter, "A genetic cascade-correlation learning algorithm," in *Proceedings of the International Workshop on Combinations of Genetic Algorithms and Neural Networks (COGANN '92)*, J. D. Schaffer and D. Whitley, Eds., pp. 123–133, Morgan Kaufmann, Baltimore, Md, USA, June 1992.
- [50] S. K. Rogers, J. M. Colombi, C. E. Martin, et al., "Neural networks for automatic target recognition," *Neural Networks*, vol. 8, no. 7-8, pp. 1153–1184, 1995.
- [51] R. J. Fogler, M. W. Koch, M. M. Moya, L. D. Hostetler, and D. R. Hush, "Feature discovery via neural networks for object recognition in SAR imagery," in *Proceedings of IEEE International Joint Conference on Neural Networks (IJCNN '92)*, vol. 4, pp. 408–413, Baltimore, Md, USA, June 1992.
- [52] H. L. Roitblat, W. W. L. Au, P. E. Nachtigall, R. Shizumura, and G. Moons, "Sonar recognition of targets embedded in sediment," *Neural Networks*, vol. 8, no. 7-8, pp. 1263–1273, 1995.
- [53] M. W. Roth, "Neural networks for extraction of weak targets in high clutter environments," *IEEE Transactions on Systems, Man and Cybernetics*, vol. 19, no. 5, pp. 1210–1217, 1989.
- [54] M. W. Roth, "Survey of neural network technology for automatic target recognition," *IEEE Transactions on Neural Networks*, vol. 1, no. 1, pp. 28–43, 1990.
- [55] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning internal representations by error propagation," in *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*,

- Volume 1: Foundations*, D. E. Rumelhart, J. L. McClelland, and the PDP Research Group, Eds., chapter 8, pp. 318–362, MIT Press, Cambridge, Mass, USA, 1986.
- [56] M. V. Shirvaikar and M. M. Trivedi, “A neural network filter to detect small targets in high clutter backgrounds,” *IEEE Transactions on Neural Networks*, vol. 6, no. 1, pp. 252–257, 1995.
 - [57] S. Singh, M. Markou, and J. Haddon, “Natural object classification using artificial neural networks,” in *Proceedings of IEEE-INNS-ENNS International Joint Conference on Neural Networks (IJCNN '00)*, vol. 3, pp. 139–144, Como, Italy, July 2000.
 - [58] F. F. Soulie, E. Viennet, and B. Lamy, “Multi-modular neural network architectures: applications in optical character and human face recognition,” *International Journal of Pattern Recognition and Artificial Intelligence*, vol. 7, no. 4, pp. 721–755, 1993.
 - [59] L. Spirkovska and M. B. Reid, “Higher-order neural networks applied to 2D and 3D object recognition,” *Machine Learning*, vol. 15, no. 2, pp. 169–199, 1994.
 - [60] C. Stahl and P. Schoppmann, “Advanced automatic target recognition for police helicopter missions,” in *Automatic Target Recognition X*, F. A. Sadjadi, Ed., vol. 4050 of *Proceedings of SPIE*, pp. 61–68, Orlando, Fla, USA, April 2000.
 - [61] B. Tian, M. R. Azimi-Sadjadi, T. H. von der Haar, and D. Reinke, “Temporal adaptive probability neural network for cloud classification from satellite imagery,” in *Proceedings of IEEE International Joint Conference on Neural Networks, IEEE World Congress on Computational Intelligence (IJCNN '98)*, vol. 3, pp. 1732–1737, Anchorage, Alaska, USA, May 1998.
 - [62] S. E. Troxel, S. K. Rogers, and M. Kabrisky, “The use of neural networks in PSRI target recognition,” in *Proceedings of IEEE International Conference on Neural Networks (ICNN '88)*, vol. 1, pp. 593–600, San Diego, Calif, USA, July 1988.
 - [63] D. Valentin, H. Abdi, and A. J. O’Toole, “Categorization and identification of human face images by neural networks: a review of the linear auto-associative and principal component approaches,” *Journal of Biological Systems*, vol. 2, no. 3, pp. 413–429, 1994.
 - [64] D. Valentin, H. Abdi, A. J. O’Toole, and G. W. Cottrell, “Connectionist models of face processing: a survey,” *Pattern Recognition*, vol. 27, no. 9, pp. 1209–1230, 1994.
 - [65] B. Verma, “A feature extraction technique in conjunction with neural network to classify cursive segmented handwritten characters,” in *Proceedings of IEEE International Joint Conference on Neural Networks, IEEE World Congress on Computational Intelligence (IJCNN '98)*, vol. 1, pp. 332–336, Anchorage, Alaska, USA, May 1998.
 - [66] B. Verma, “A neural network based technique to locate and classify microcalcifications in digital mammograms,” in *Proceedings of IEEE International Joint Conference on Neural Networks, IEEE World Congress on Computational Intelligence (IJCNN '98)*, vol. 3, pp. 1790–1793, Anchorage, Alaska, USA, May 1998.
 - [67] A. M. Waxman, M. C. Seibert, A. Gove, et al., “Neural processing of targets in visible, multispectral IR and SAR imagery,” *Neural Networks*, vol. 8, no. 7-8, pp. 1029–1051, 1995.
 - [68] T. Wessels and C. W. Omlin, “A hybrid system for signature verification,” in *Proceedings of IEEE-INNS-ENNS International Joint Conference on Neural Networks (IJCNN '00)*, vol. 5, pp. 509–514, Como, Italy, July 2000.
 - [69] D. Whitley and T. Hanson, “Optimizing neural networks using faster, more accurate genetic search,” in *Proceedings of the 3rd International Conference on Genetic Algorithms (ICGA '89)*, pp. 391–396, Morgan Kaufmann, Fairfax, Va, USA, June 1989.
 - [70] P. Winter, S. Sokhansanj, H. C. Wood, and W. Crerar, “Quality assessment and grading of lentils using machine vision,” in *Agricultural Institute of Canada Annual Conference*, Canadian Society of Agricultural Engineering, Saskatoon, Canada, July 1996, CASE paper no. 96-310. 222.
 - [71] P. Winter, W. Yang, S. Sokhansanj, and H. Wood, “Discrimination of hard-to-pop popcorn kernels by machine vision and neural network,” in *Proceedings of the ASAE/CSAE North-Central Intersectional Meeting*, Saskatoon, Canada, September 1996, paper no. MANSASK 96-107.
 - [72] Y. Won, J. Nam, and B.-H. Lee, “Image pattern recognition in natural environment using morphological feature extraction,” in *Proceedings of Joint IAPR International Workshops on Syntactical*

- and Structural Pattern Recognition and Statistical Pattern Recognition (SSPR-SPR '00)*, F. J. Ferri, J. M. Iñesta, A. Amin, and P. Pudil, Eds., vol. 1876 of *Lecture Notes in Computer Science*, pp. 806–815, Springer, Alicante, Spain, August-September 2000.
- [73] Y. C. Wong and M. K. Sundareshan, “Data fusion and tracking of complex target maneuvers with a simplex-trained neural network-based architecture,” in *Proceedings of IEEE International Joint Conference on Neural Networks, IEEE World Congress on Computational Intelligence (IJCNN '98)*, vol. 2, pp. 1024–1029, Anchorage, Alaska, USA, May 1998.
 - [74] X. Yao, “A review of evolutionary artificial neural networks,” *International Journal of Intelligent Systems*, vol. 8, no. 4, pp. 539–567, 1993.
 - [75] X. Yao, “Evolving artificial neural networks,” *Proceedings of the IEEE*, vol. 87, no. 9, pp. 1423–1447, 1999.
 - [76] X. Yao and Y. Liu, “A new evolutionary system for evolving artificial neural networks,” *IEEE Transactions on Neural Networks*, vol. 8, no. 3, pp. 694–713, 1997.
 - [77] A. Ylä-Jääski and F. Ade, “Grouping symmetrical structures for object segmentation and description,” *Computer Vision and Image Understanding*, vol. 63, no. 3, pp. 399–417, 1996.
 - [78] M. Zhang, *A domain independent approach to 2D object detection based on the neural and genetic paradigms*, Ph.D. thesis, Department of Computer Science, RMIT University, Melbourne, Australia, August 2000.
 - [79] M. Zhang and V. Ciesielski, “Neural networks and genetic algorithms for domain independent multiclass object detection,” *International Journal of Computational Intelligence and Applications*, vol. 4, no. 1, pp. 77–108, 2004.

Mengjie Zhang: School of Mathematics, Statistics and Computer Science, Victoria University of Wellington, P.O. Box 600, Wellington 6140, New Zealand

Email: mengjie.zhang@mcs.vuw.ac.nz

21

An evolutionary approach for designing multitarget tracking video systems

Jesús García, Óscar Pérez, Antonio Berlanga, and José M. Molina

21.1. Introduction

A video surveillance system is usually made up of several blocks that carry out interrelated processes at different levels: characterization of background, detection of moving pixels, grouping moving pixels (blobs), association of blobs to real targets, tracks parameters estimation, and so forth [16]. Thus, each of these blocks has a set of adjustable parameters that regulate the different processes performing machine-vision tasks, and as a result of this, each one of them rules the performance of part of a particular block.

A new generation of surveillance systems has developed new technologies based on the acquisition and analysis in real time of digital images. Digital technology has not changed the essence of the surveillance and tracking operations but has extended the reliability and the wide range of possibilities to analyse the information. Moreover, such technologies are in an experimental stage and so, many proposals are being suggested in different surveillance applications [15, 17, 23].

In our case, we have the challenge of building a specific video surveillance system which is focused on the detection and tracking of moving objects [1, 11].

The purpose of this work consists of obtaining the most general set of parameters by means of evolution strategies, that is, the parameters which allow the surveillance system to have the best performance in different scenarios and conditions.

Thus, our work proposes a machine-vision application built and adjusted by means of a collection of examples or scenarios of the real world. These examples must represent the widest range of different situations that might happen so that the system is trained and fixed for a general set of conditions [24].

Specifically, many of the machine-vision systems are focused on surveillance.

In order to achieve our purpose, we propose several steps to follow. First of all, we take a representative set of scenarios to train the system. In particular, our work is focused on an airport domain and all the scenarios represent common situations for surface movements. This application of video technology in airport

areas is a new way to support ground traffic management inside the new paradigm of or advanced surface movement, guidance, and control systems (ASMGCS) [2, 14, 22]. The surveillance system used for this study is based on some previous works carried out in the airport domain [6].

Second, evolution strategies are the technique chosen to adjust the parameters that give the best performance to the tracking system [25, 28, 32]. One of the innovations of our proposal is that the fitness function is adjusted in different ways in order to optimize the tracking of distinct moving multitargets, from a single one in a specific scenario to many of them in different scenarios. That is the process followed to obtain the most general set of parameters. Thus, whereas *overfitting* means obtaining particular parameters for specific situations, *generalization* is the capability to adapt the parameters to be used in many different cases [33, 34].

The reason for the election of ES technique for this application is justified since the function landscape is very irregular, with plenty of local optima. Classical techniques of optimization such as those based on a gradient descent are poorly suitable to these types of problems, when there is a high number of local optima, or other properties like high roughness and local flat surfaces [3, 5].

In the next section, the video processing system is outlined, indicating the structure and effects of parameters open to be adjusted by the designer. The third section describes the scenarios taken into account in the design, as a sample of the typical characteristics in this airport surveillance domain. The fourth section presents the evaluation procedure, suggested metrics, and aggregation scheme for including multiple situations. Finally, the fifth section presents the optimization results carried out to assess the effectiveness of the proposal, closing the work with some final conclusions.

21.2. Surveillance video system

This section describes the structure of an image-based tracking system. Camera sensors are being explored as a complementary source of data in this environment [9], with respect to classical sensors such as surface movement radars [31].

The system architecture is a coupled tracking system where the detected objects are processed to initiate and maintain tracks (see Figure 21.1). These tracks represent the real targets in the scenario and the system estimates their location and cinematic state. The detected pixels are connected to form image regions referred to as blobs.

The association process assigns one or several blobs to each track, while nonassociated blobs are used to initiate tracks.

It is important to point out that the adjustment parameters that are optimized are written in uppercase.

21.2.1. Detector and blobs extraction

The positioning/tracking algorithm is based on the detection of moving multiple targets by contrasting with local background [10], whose statistics are estimated

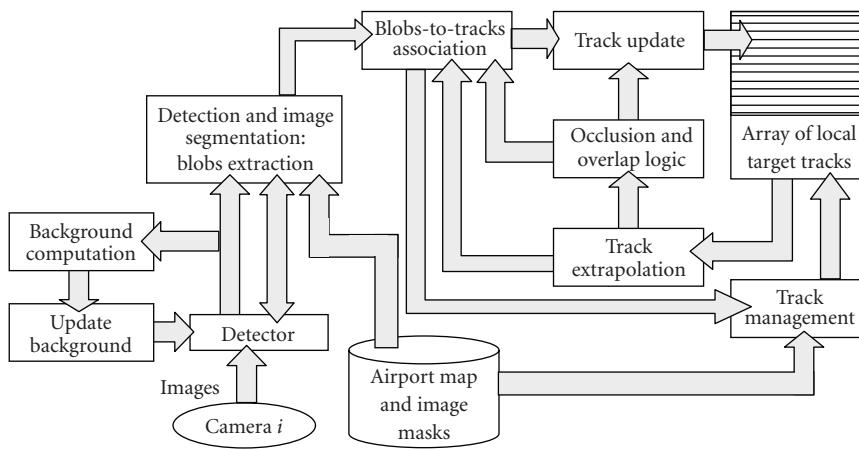


FIGURE 21.1. Structure of the video surveillance system.

and updated with the video sequence. Then, the pixel level detector is able to extract moving features from background, comparing the difference with a threshold:

$$\text{Detection}(x, y) := [\text{Image}(x, y) - \text{Background}(x, y)] > \text{THRESHOLD}^* \sigma \quad (21.1)$$

being σ the standard deviation of pixel intensity. This parameter determines the first filter on the data amount to be processed in following phases [29].

Finally, the algorithm for blobs extraction marks with a unique label all moving detected pixels, by means of a clustering and growing regions algorithm [30]. Then, the rectangles which enclose the resulting blobs are built, and their centroids and areas are computed. In order to reduce the number of false detections due to noise, a minimum area, MIN_AREA , is required to form blobs. This parameter is a second data filter which avoids noisy detections from the processing chain.

21.2.2. Blobs-to-track association

The association problem lies in deciding the most proper grouping of blobs and assigning them to each track for each frame processed [8, 12, 21]. Due to image irregularities, shadows, occlusions, and so forth, a first problem of imperfect image segmentation appears, resulting in multiple blobs generated for a single target. So, the blobs must be reconnected before track assignment and updating. However, when multiple targets move closely, their image regions may overlap. As a result, some targets may appear occluded by other targets or obstacles, and some blobs can be shared by different tracks. For the sake of simplicity, first a rectangular box has been used to represent the target. Around the predicted position, a rectangular

box with the estimated target dimensions is defined, $(x_{\min}, x_{\max}, y_{\min}, y_{\max})$. Then, an outer gate, computed with a parameter defined as a margin, **MARGIN_GATE**, is defined. It represents a permissible area in which to search more blobs, allowing some freedom to adapt target size and shape.

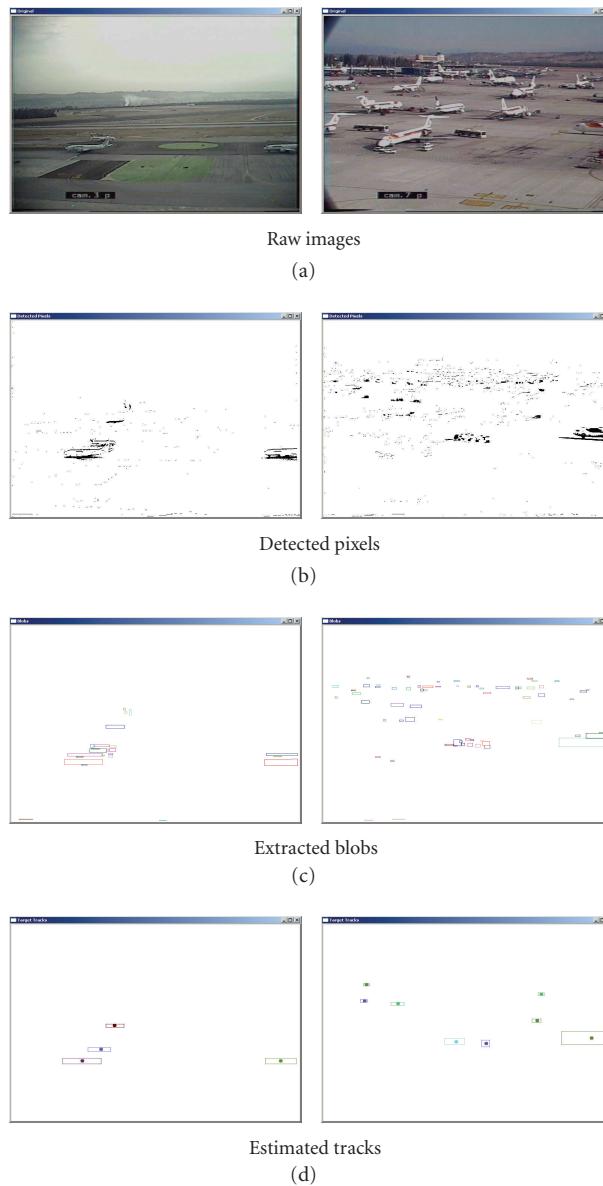
The association algorithm analyses the track-to-blob correspondence [18, 20]. It first checks if the blob and the track rectangular gates are compatible (overlap), and marks as conflictive those blobs which are compatible with two or more different tracks. After gating, a grouping algorithm is used to obtain one “pseudoblob” for each track. This pseudoblob will be used to update track state. If there is only one blob associated to the track and the track is not in conflict, the pseudoblob used to update the local track will be this blob [27]. Otherwise, two cases may occur.

- (1) A conflict situation arises when there are overlapping regions for several targets (conflicting tracks). In this case, the system may discard those blobs gated by several tracks and extrapolate the affected tracks. However, this policy may be too much restrictive and might degrade tracking accuracy. As a result, it has been left open to design by means of a Boolean parameter named **CONFLICT** which determines the extrapolation or not of the tracks.
- (2) When a track is not in conflict, and it has several blobs associated to it, these will be merged on a pseudoblob whose bounding limits are the outer limits of all associated blobs. If the group of compatible blobs is too big and not dense enough, some blobs (those which are further away from the centroid) are removed from the list until density and size constraints are held. The group density is compared with a threshold, **MINIMUM.DENSITY**, and the pseudoblob is split back into the original blobs when it is below the threshold.

21.2.3. Tracks filtering, initiation, and deletion

A recursive filter updates centroid position, rectangle bounds, and velocity for each track from the sequence of assigned values, by means of a decoupled Kalman filter for each Cartesian coordinate, with a piecewise constant white acceleration model. The acceleration variance that will be evaluated, usually named as “plant-noise,” is directly related with tracking accuracy. The predicted rectangular gate, with its search area around, is used for gating. Thus, it is important that the filter is “locked” to real trajectory. Otherwise, tracks would lose its real blobs and finally drop. So this value must be high enough to allow manoeuvres and projection changes, but not too much, in order to avoid noise. As a result, it is left as an open parameter to be tuned, **VARIANCE.ACCEL**.

Finally, tracking initialization and management takes blobs which are not associated to any previous track. It requires that nongated blobs extracted in successive frames accomplish certain properties such as a maximum velocity and similar sizes which must be higher than a minimum value established by the parameter **MINIMUM.TRACK.AREA**. In order to avoid multiple splits of targets,

FIGURE 21.2. Information levels in the processing *chain*.

established tracks preclude the initialization of potential tracks in the surrounding areas, using a different margin than the one used in the gating search. This value which allows track initialization is named MARGIN_INITIALIZATION.

To illustrate the whole process, Figure 21.2 depicts the different levels of information interchanged, from the raw images until the final tracks.

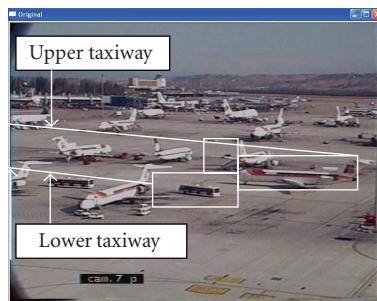


FIGURE 21.3. Picture of the first scenario.

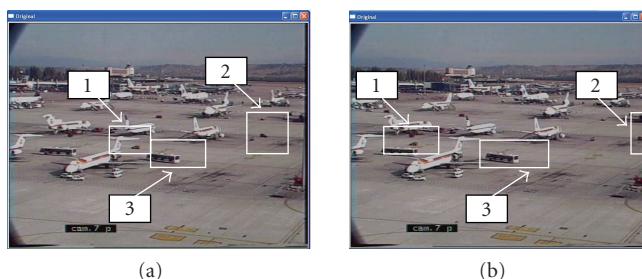


FIGURE 21.4. Picture of the first scenario for the targets number 1, 2, and 3 at the beginning of the scenario (a) and after 7 seconds (b).

21.3. Set of examples

This section shows the set of three types of scenarios that we have used for our experiments and the selected objects for tracking. The scenarios represent a good set for training the system as they are long and varied enough to cover the most common situations of surface movements in an airport.

(i) The first scenario is a multipleblob reconnection scenario. An aircraft is moving from left to right with partial appearance because it is hidden by other aircraft and vehicles parked in the parking places. Thus, there are multiple blobs which represent this aircraft that must be reconnected. At the same time, there are four vehicles (three cars and a bus) moving on parallel roads or inner taxiways (upper taxiway and lower taxiway) whose tracks must be kept separated from this aircraft trajectory.

Figure 21.4 and 21.5 show the enumeration followed by the program to identify each target. As we said, the evaluation is carried out over each trajectory on moving object, so these numbers will be useful to identify the targets in the tables of results.

This first scenario contains five targets as follows.

- (1) A car which goes from the centre to the left side of the screen following the lower taxiway.

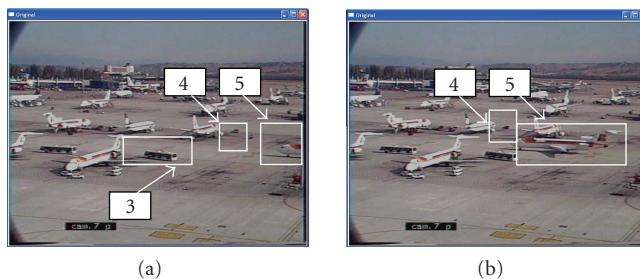


FIGURE 21.5. Picture of the first scenario for the targets number 4 and 5 after 19 seconds (a) and 25 seconds (b).

- (2) A car which heads towards the right side of the picture in the upper taxiway.
- (3) A bus for passengers. This bus makes very slow and slight movements, practically nonappreciable.
- (4) Another car that appears in the right side of the pictures, drives all along the upper taxiway, and finally disappears in the left side.
- (5) And finally, a big aircraft that heads from the left to right in the lower taxiway.

Figure 21.3 shows the sequence of movements in the first scenario.

- (ii) The second scenario presents three aircraft moving in parallel taxiways (see Figure 21.6). The aircraft images overlap when they are crossing. This always occurs with uniform motion on straight segments (Figures 21.7 and 21.8).
- (iii) Finally, the third scenario presents two aircrafts moving on inner taxiways between airport parking positions. The aircraft moves in different directions. The first conflict situation arises when one of the aircrafts turns to the right and it is partially hidden by the other aircraft, which has already changed its direction by turning to the left. The second conflict situation happens at the end of the video sequence when another airplane appears on the right side of the taxiway. This new airplane is also partially hidden by the first aircraft that is disappearing from the field of view of the camera in that moment.

Finally, this third scenario presents also three targets.

- (1) The first target is an aircraft that appears in the left side, turns to the left and has a conflicting situation with the second target (another aircraft) that is moving in a parallel taxiway. We say that is a conflicting situation because it will be difficult for the tracking system to separate both targets.
- (2) The second aircraft appears in the left side of the screen, turns to the right and before disappearing, it is crossing with another aircraft (the third target) that suddenly appears. This will be a conflicting situation and the detector will not be able to distinguish separately the two targets.



FIGURE 21.6. Picture of the second scenario.

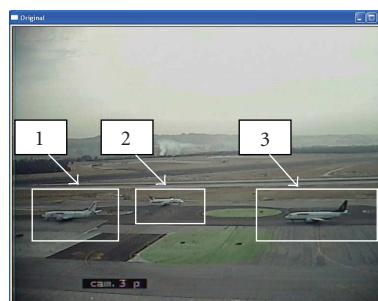


FIGURE 21.7. Picture of the second scenario for the targets number 2, 3, and 4 after 26 seconds.

- (3) As we said above, the third target is another aircraft that appears in the right side at the end of the video.

Figure 21.9 shows the moving targets in the second 25.

21.4. Adjustment of surveillance system: evaluation and generalization

The method presented in this section carries out the search for suitable parameters for the best performance of the surveillance video system. The pursued goals of this method are as follows.

- (1) Evaluation of the tracks provided by the surveillance system. Thus, we are going to assess the performance of the surveillance video system by evaluating *each* resulting single track and by means of some proposed evaluation metrics.
- (2) Adaptation and optimization (by means of evolution strategies) of the parameters that regulate the system processes. It is necessary the whole adaptation of these processes taking into account that they are strongly interrelated.
- (3) Generalization, understood as the capacity of the parameters found to solve several cases, adapting the system for a whole group of situations.



FIGURE 21.8. Picture of the second scenario.



FIGURE 21.9. Picture of the third scenario for the targets number 1 and 2 after 25 seconds.

The approach is depicted in Figure 21.10 by means of a loop that will be stopped after accomplishing a condition that fulfill the requirement of obtaining a suitable set of parameters for the surveillance system. Thus, the definition of an evaluation methodology allows automating the search of these parameters and adjusting the whole system to the best performance in the considered cases.

The triple arrow in Figure 21.10 represents that the surveillance detections of a specific track and its correspondent reference data (ground truth) are introduced in the evaluation module for their assessment and subsequent adjustment of the parameters.

21.4.1. Evaluation metrics and evaluation function

The evaluation function calculates a numerical value that represents the quality of the tracking system with regard to a reference track or ground truth [7]. Two aspects have been considered.

(i) The ground truth is the result of a study from pre-recorded video sequences and a subsequent process in which a human operator selects coordinates for each target [13, 26, 27]. The coordinates of the targets are selected frame by frame; they are marked and bounded with rectangles, taking the upper-left and lower-right corners as location of target objectives. Thus, the ground truth can be

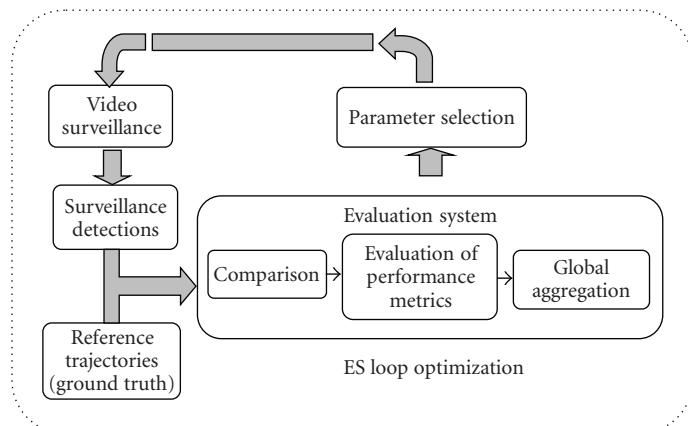


FIGURE 21.10. Approach for system adjustment.

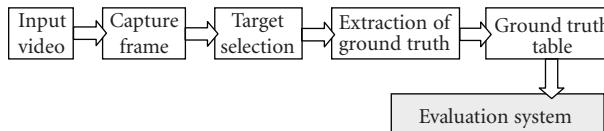


FIGURE 21.11. Extraction of the ground truth.

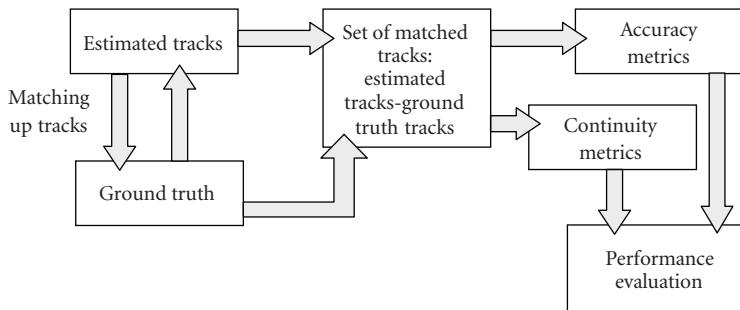
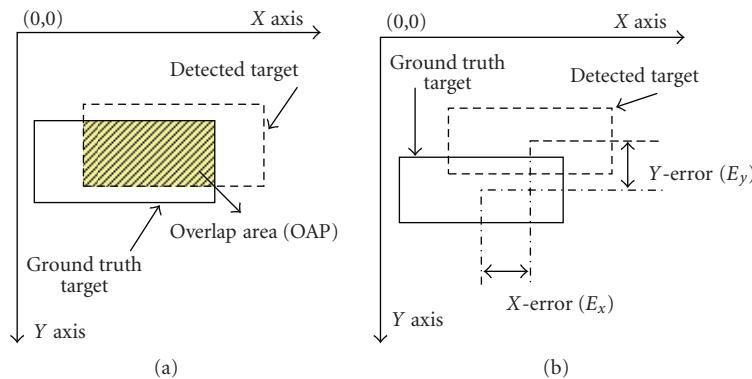


FIGURE 21.12. Computation of evaluation metrics.

defined as a set of rectangles that define the real locations of each target. Finally, they are stored in a table which will be used to compare with the output tracks of the tracking system. Figures 21.11 and 21.12 show the extraction and evaluation processes, respectively.

(ii) The performance evaluation system calculates some numerical values by means of a set of proposed metrics, based on the ground truth mentioned above.

FIGURE 21.13. Metrics of overlap area, X -error and Y -error.

For the evaluation, the output tracks must be matched with ground truth trajectories in the first place. The optimization of this evaluation outcome will be the goal for the evolutionary strategy program.

The evaluation system computes four parameters per target which are classified into “accuracy metrics” and “continuity metrics.”

Accuracy metrics (see Figure 21.13).

- (1) *Overlap-area (OAP):* overlap area (in percentage) between the real target and the detected track.
- (2) *X-error (E_x) and Y-error (E_y):* absolute difference, in x and y coordinates, between their centres.

Continuity metrics.

- (3) *Number of Tracks per target (NT):* it is checked if more than one detected track is matched with the same ideal track. If this happens, the program keeps the detected track which has a bigger overlapped area value, removes the other one and marks the frame with a flag that indicates the number of detected tracks associated to this ideal one.
- (4) *Commutation (C):* a commutation occurs (see Figure 21.14) when the identifier of a track matched to an ideal track changes. It typically takes place when the track is lost and recovered later.

The evaluation function is based on the previous metrics, by means of a weighted sum of different terms which are computed for each target:

$$E = \frac{W_1 M}{T} + \frac{W_2 \sum (1 - OAP) + W_3 \sum E_x + W_4 \sum E_y}{CT} + \frac{W_5 O_C + W_6 U_C + W_7 \sum C}{T} \quad (21.2)$$

with the terms defined as follows.

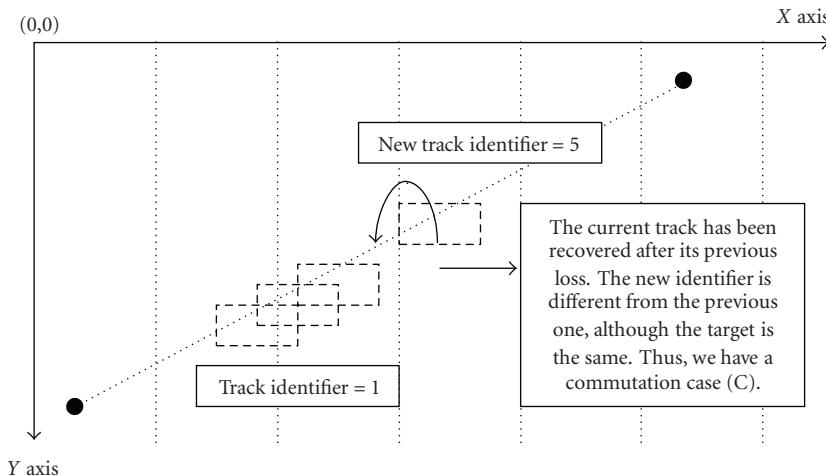


FIGURE 21.14. Commutation metric.

- (i) Mismatch (M): a counter which stores how many times the ground truth and the tracked object data do not match up.
- (ii) The three next terms are the total sum of the nonoverlapped areas ($\sum(1 - \text{OAP})$) and the central error of x ($\sum E_x$) and y axes ($\sum E_y$).
- (iii) The next two elements are two counters.
 - (a) Overmatch-counter (O_c): how many times the ground truth track is matched with more than one track object data.
 - (b) Undermatch-counter (U_c): how many times the ground track is not matched with any track at all.
- (iv) The number of commutations in the track under study ($\sum C$).
- (v) The continuity elements are normalized by the time length of track, T , while the accuracy terms are normalized by the time length of track being continuous, CT (i.e., when they can be computed).
- (vi) $W_{1,2,3,4,5,6,7}$ are the relative weights for the terms (detailed later). Highest values have been given to the continuity terms, since this aspect is the key to guarantee the global viability.

21.4.2. Fitness aggregation over several trajectories

We have presented above the evaluation function per target. In order to carry out a general evaluation over different targets and cases, aggregation operators must be applied over partial evaluations. The initial or basic function is this evaluation function per target (or track):

$$e_{ij} = f(x_{ij}^-, \theta) = \frac{W_1 M}{T} + \frac{W_2 \sum (1 - \text{OAP}) + W_3 \sum E_x + W_4 \sum E_y}{CT} + \frac{W_5 O_c + W_6 U_c + W_7 \sum C}{T}, \quad (21.3)$$

where:

- (a) e_{ij} is the evaluation result for the i th track/target in the scenario j th scenario;
- (b) x_{ij}^- is the vector of metrics and;
- (c) θ is the vector of parameters to optimize.

Thus, the extension of the evaluation function must allow assessing simultaneously:

- (a) one or various targets per scenario:

Scenario j : $\{e_{1j}, e_{2j}, e_{3j}, \dots, e_{N_j j}\}$;

- (b) various scenarios with several targets per scenario:

M Scenarios: $\{e_{11}, e_{21}, e_{31}, \dots, e_{N_1 1}, \dots, e_{1j}, e_{2j}, e_{3j}, \dots, e_{N_j j}, \dots, e_{1M}, e_{2M}, e_{3M}, \dots, e_{N_M, M}\}$.

Two aggregation operators have been analysed.

- (a) Sum:

$$E_j = \sum_i e_{ij}, \quad E = \sum_i \sum_j e_{ij}. \quad (21.4)$$

- (b) Maximum (or minimax):

$$E_j = \max_i (e_{ij}); \quad E = \max_i \left(\max_j (e_{ij}) \right). \quad (21.5)$$

In fact, the maximum operator generally has a good performance for aggregation in multiobjective optimization problems with a complex tradeoff required, as it was showed in a previous work related with optimization applied to a multiconstraint design problem [19]. The sum operator has been taken as a reference for comparison.

21.5. Results

This section shows the approached evaluation method, and the subsequent use of ES optimization, for the video tracking system. The eight parameters pointed before are searched for the best performance of the surveillance system under different situations. In order to analyse the generalization capability, this search is done in three steps, from the particular optimization of a single target (the parameters for the best tracking of a single target) to the most general optimization (the parameters for the best tracking of all targets of all videos).

- (i) First, the parameters will be optimized to obtain the best tracking performance for specific targets in every scenario. This means that the system finds the best parameters for each specific track, and it will be the benchmark to compare the rest of results. Thus, the number of parameter sets will be the same as the number of targets in all videos.

- (ii) Second, the best parameters will be searched for the best tracking performance for the set of targets composing every scenario. In order to be able to compute the fitness value of one scenario, an aggregation operator of the individual fitness values of each target must be used as fitness function. As it was said before, sum and maximum operators were applied to compute the total fitness for each scenario. Thus, the number of parameter sets will be the same as the number of videos or scenarios.
- (iii) Finally, the optimization will be applied to obtain the best performance for the whole set of scenarios. In this section, we are not going to focus on a specific target, not even on a specific scenario, but on the whole set of scenarios. As it was said in the previous section, an aggregation operator of the individual fitness values of each target must be used to compute the total fitness function for the set of videos. We also use the sum and maximum aggregation functions as fitness function.

21.5.1. Implementation details: solution encoding, evaluation of tracking performance and system optimization

As it was mentioned before, the evolution strategies are the technique applied to optimize the system performance. The fitness function is defined over the system output compared with references, accordingly to (21.2), and applying aggregation operators over partial evaluations to the set of trajectories considered to adapt the system, following (21.3). Since this function comprises error terms, the lower the evaluation function, the better the quality of the tracking system.

Regarding implementation, the size of population was reduced to the minimum one in order to carry out the experiments within a reasonable time, while convergence to appropriate solutions was kept with an appropriate setting for mutation factor. The configuration of ES optimization is outlined below.

- (a) Continuous representation, a solution is a vector in \mathbb{R}^8 , with all components directly encoding the configuration parameters highlighted in Section 21.2.
- (b) Each individual comprises 8 real numbers, all constrained to be positive: (THRESHOLD, MINIMUM_AREA, MARGIN_GATE, MINIMUM_DENSITY, CONFLICT, VARIANCE_ACCEL, MINIMUM_TRACK_AREA, MARGIN_INITIALIZATION).
- (c) Selection scheme $(\mu + \lambda)$ -ES, applying discrete crossover.
- (d) Size of population and offspring = 6 + 6 ($\mu = \lambda = 6$).
- (e) Gaussian mutation, where the adaptation factor $\Delta\sigma$ was tuned to 0.5.
- (f) Stopping criterion, when the fitness function gets stagnated between consecutive generations or a maximum number of generations of 200 is achieved (it was checked that the algorithm almost always had converged before the 200 generations).

Regarding the evaluation function in (21.2), (21.3), the vector of weights was $(W_1, W_2, W_3, W_4, W_5, W_6, W_7) = (10^4, 1, 1, 1, 10^4, 5 * 10^3, 10^4)$. Very high values were given to the continuity terms to enforce a robust behavior.

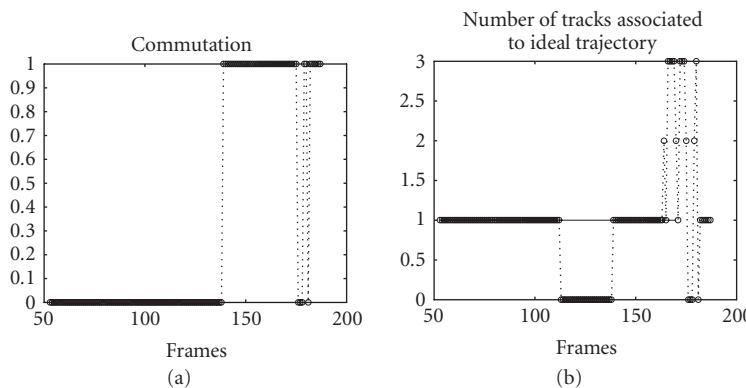


FIGURE 21.15. Direct optimization with a sample trajectory, continuity metrics: “—” optimized parameters, “o· · · o” initial parameters.

As an example of the tracking evaluation and direct optimization, Figures 21.15 and 21.16 present the tracking performance metrics, before and after optimization, taking a single-trajectory evaluation as a direct fitness function. This result corresponds to one of the objects (the big aircraft, target 5) in the first scenario described, see Figure 21.5. As we can see, the system presents track losses and commutations with an initial random set of parameters, effects that disappear after the optimization. Significantly, the X-Y errors and overlap degree between tracking and ground truth also are apparently improved. These errors can be measured only in situation of continuity (one-to-one correspondence with the ideal trajectory).

21.5.2. Benchmark table: optimization over a single target (individual optimization)

The optimization over single tracks, presented in previous paragraph, has been applied to every track in the three scenarios. The results are showed in Table 21.1. This table is taken as the reference to compare with next experiments, which will apply the parameter optimization process over combinations of several targets/tracks simultaneously. The columns represent the target/track whose parameters are optimized to obtain the best performance and the best fitness function. The rows are the cross-evaluation over every target in each video. For each scenario, a sum row displays at the end the total error for the targets contained, applying the solution corresponding to that column.

Thus, the values in the diagonal (grey-shaded values) represent the overfitted cases, which means that the evaluation is carried out with the parameters particularly adjusted to this case. That is the reason why this value is the minimum in each column.

From now on, in some particular cases, the output parameters are not capable to detect the evaluated target (the undermatch-counter adds 1 unit to itself

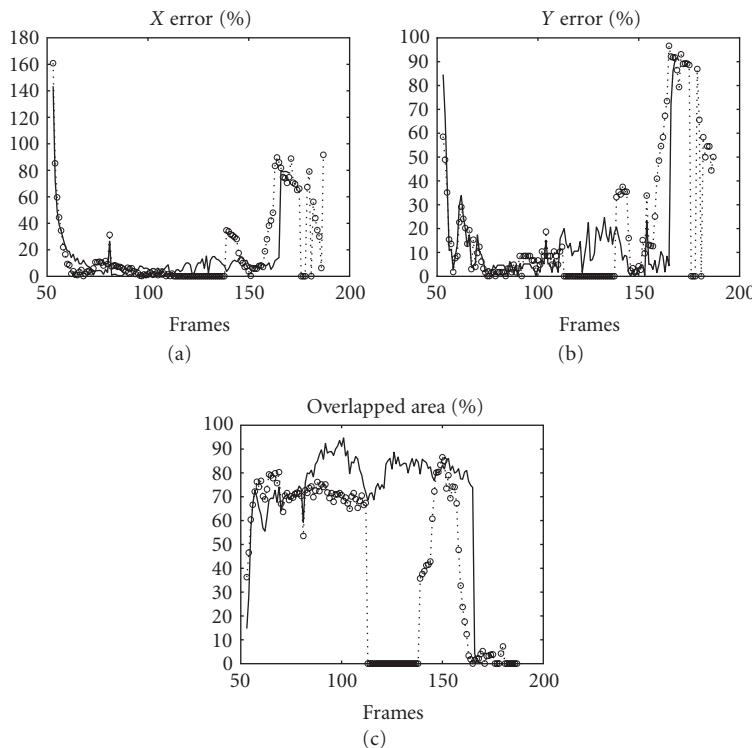


FIGURE 21.16. Direct optimization with a sample trajectory, accuracy metrics: “—” optimized parameters, “o · · · o” initial parameters.

(U_c)), which is marked with the value 10,000 (a very high weigh indicating that the evaluation gives a low performance).

In order to illustrate the overfitting effect in the results, we have selected 3 solutions from the 11 optimized sets of parameters. One target was selected per scenario, highlighted in the previous table:

- (i) target number 1 from video 1: param-Video1-T1;
- (ii) target number 1 from video 2: param-Video2-T1;
- (iii) target number 2 from video 3: param-Video3-T2.

So, in Figure 21.17 we can see three sets of columns, representing the performance of each solution when it is evaluated against the 11 targets. We can check that the parameters that optimize the track of target 1 (T1) in Video1 (param-Video1-T1) have the best cross evaluation value for the targets that belong to Video1 (the five first targets). In the same way, the parameters that optimize the track of target 1 (T1) in Video2 (param-Video2-T1) have the best cross evaluation value for the targets that belong to Video2 and worse in the rest of scenarios; and so on for the parameters in the optimization for a target belonging to Video3.

In order to have a reference figure to be compared with the subsequent experiments, Figure 21.18 shows the average of all evaluations with the optimum

TABLE 21.1. Cross evaluation of the optimized parameters for each single track.

evaluation scenario	designed scenario	Video 1			Video 2			Video 3			Average
		param-Video1-T1	param-Video1-T2	param-Video1-T3	param-Video1-T4	param-Video1-T5	param-Video2-T1	param-Video2-T2	param-Video2-T3	param-Video3-T1	
Video 1	Video1-T1	1679,9	10493	10010	10000	5252,1	17140	8389,9	5851,7	2309,1	3068,1
	Video1-T2	2816,2	2834,9	2838,5	2837,9	2838,5	2837,8	2836,9	2838,0	2838,2	2837,4
	Video1-T3	1574,1	7686,8	4102,2	978,3	7987,1	7802,9	7569,4	6350,4	7486,8	12602
	Video1-T4	3753,0	5867,0	4106,5	257,9	4947,8	7100,0	4772,6	5155,3	3596,9	3625,1
	Video1-T5	221,4	6581,2	5451,0	10000	47,6	7250,1	5141,1	7293,3	156,4	4869,6
	Sum1	10063	33444	26508	24074,2	21058,2	42131	28710	27487	16387	26943
Video 2	Video2-T1	1244,2	6879,2	493,5	2501,3	572,8	248,9	498,1	470,7	3757,2	529,7
	Video2-T2	9091,7	8209,0	1343,0	6238,2	2393,5	1510,0	731,4	1521,7	6765,8	6905,9
	Video2-T3	2946,6	7946,6	759,8	768,7	1496,9	752,6	770,4	743,1	786,1	761,8
	Sum2	13282	23034	2596,2	9508,1	4463,1	2511,6	1999,9	2735,4	11309	8197,5
	Sum3										8357,5
Video 3	Video3-T1	2720,7	7029,5	9959,7	583,1	57546,1	6508,1	5862,6	6157,4	172,5	6756,8
	Video3-T2	7706,9	8104,1	13271	305,5	10189,5	9771,7	5080,6	12383	5079,4	91,2
	Video3-T3	11388	9460,7	12319	10000	14943,6	14421	11711	16524	9445,9	13398
	Sum3	21815	24594	35550	10888,6	82679,2	30701	22654	35065	14697	20246
Total Sum (Sum1 + Sum2 + Sum3)		45161	81073	64655	44470	108200	75344	53365	65288	42394	55387
Faults		0	0	0	2	0	0	0	0	0	1

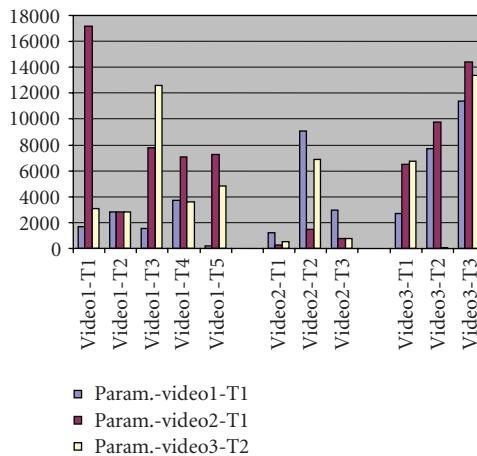


FIGURE 21.17. Evaluation of the obtained parameters over particular trajectories.

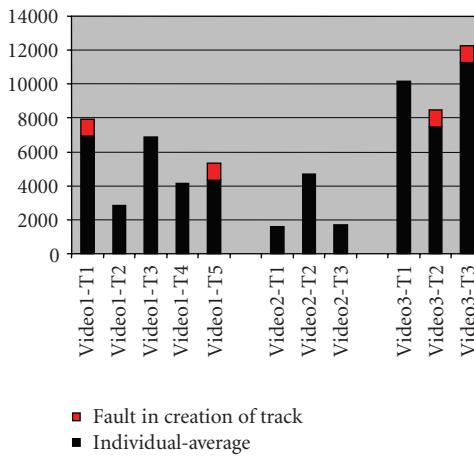


FIGURE 21.18. Average of evaluations over the parameters for each target.

parameters for each target. They correspond to the last column in Table 21.1: each cell is the average of the cells in that row. In some way, these values represent the overfitting effect since each row contains the particular optimum value when the same target is used for optimization, and other ten cases corresponding to particular optimizations for other targets.

21.5.3. Optimization over targets of a specific video: scenario optimization

The next step is the optimization for all the tracks included in a video, applying the addition and the maximum operators. The evaluation is carried out over all

TABLE 21.2. Cross evaluation for best parameters per scenario with minimax operator.

	designed scenario	Video 1	Video 2	Video 3	Average
evaluation scenario		param-Video1	param-Video2	param-Video3	
Video 1	Video1-T1	2148,38	6467,07	7118,28	5244,57
	Video1-T2	2816,06	2838,03	2829,93	2828,00
	Video1-T3	808,49	7571,34	7722,86	5367,56
	Video1-T4	611,94	4296,65	2346,35	2418,31
	Video1-T5	219,49	6012,59	6898,39	4376,82
	Sum1	6604,36	27185,68	26915,81	20235,28
Video 2	Video2-T1	7761,07	501,94	498,34	2920,45
	Video2-T2	4774,85	735,06	1709,53	2406,48
	Video2-T3	5057,37	746,73	1901,62	2568,57
	Sum2	17593,2	1983,73	4109,49	7895,50
Video 3	Video3-T1	327,80	8511,76	1321,73	3387,09
	Video3-T2	352,39	6639,13	4445,21	3812,24
	Video3-T3	10000	10007,40	2724,84	7577,41
	Sum3	10680,19	25158,29	8491,78	14776,75
Total Sum (Sum1 + Sum2 + Sum3)		34877,84	54327,70	39517,08	
Faults		1	0	0	

the trajectories per video. Subsequently, both operators are applied, minimization of the maximum (minimax) and minimization of the sum, in order to obtain the value that the evolution strategy algorithm uses to search for better individuals.

Table 21.2 shows the results for the maximum operator. There are three columns which correspond with the optimization carried out over each scenario or video. The overfit effect can be checked again in the diagonal of the table (grey-shaded values). These values are the evaluation over the tracks used in the training process.

Following this method (Table 21.2), we obtained a fitness value which is remarkably better than all previous results, which were calculated by training over a single track (Table 21.1). For example, for the first video, the sum of the fitness gives a value of 6604.36. If we compare to the different sums given before for each previous solution in Table 21.1 (10063.237, 33444.39, 26508.18, and 21058.22), it is easy to infer the improvement of the performance. The same argument can be followed for the rest of videos. In video number 2, the sum of fitness is 1983.73

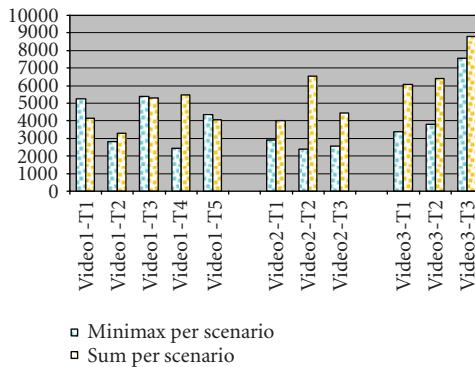


FIGURE 21.19. Comparison of the evaluations carried out with the maximum or addition operator over all the tracks that belong to a specific video.

while the values obtained in the first experiment were 2511.56, 1999.89, and 2735.41.

In an analogous way, the sum operator was applied later in order to obtain the input for the evolution strategy algorithm.

The values obtained by the sum operator show worse performance of the tracking system than previous case, as it can be observed in Figure 21.19, comparing both operators. This figure shows the evaluation average over each track for the three experiments (last column of Tables 21.2, 21.3), instead of comparing the particular solutions obtained for each one of the three scenarios.

21.5.4. Optimization over targets belonging to the whole set of videos: global optimization

The final step searches for the best set of parameters applied to all targets in all videos. As in the previous experiment, we will use the addition of every single fitness values (the results of the evaluation of the set of parameters over every single target) and the maximum among all the single fitness values.

The results are shown in Tables 21.4 and 21.5. The best aspect of this global optimization lies in the validity of the parameters to the whole set of videos, avoiding the overfitted values obtained in the previous experiments. Table 21.4 shows the results for the case in which the maximum fitness has been taken in each iteration loop (the strategy of maintaining the maximum of the evaluation values is called the “minimax” strategy). The results applied to a specific scenario are good, but not better than the results obtained in the previous section where with the particular parameters obtained for that situation.

Nevertheless, the total fitness sum for all the scenarios results much better considering evaluations over all situations. This fact indicates that the parameters fit well in the general case.

TABLE 21.3. Cross evaluation for the best parameters per scenario with sum operator.

	designed scenario	Video 1	Video 2	Video 3	Average
evaluation scenario		param-Video1	param-Video2	param-Video3	
Video 1	Video1-T1	1691,76	2650,92	8046,10	4129,59
	Video1-T2	2843,30	2842,96	4232,38	3306,21
	Video1-T3	1141,46	1684,53	13112,50	5312,83
	Video1-T4	541,82	10000	5826,94	5456,25
	Video1-T5	177,96	10000	2024,31	4067,42
Sum1		6396,30	27178,41	33242,23	22272,31
Video 2	Video2-T1	1744,78	284,92	10000	4009,9
	Video2-T2	9155,52	501,81	10000	6552,44
	Video2-T3	4403,98	784,92	8095,01	4427,97
Sum2		15304,28	1571,64	28095,01	14990,31
Video 3	Video3-T1	7608,18	10000	631,79	6079,99
	Video3-T2	6906,09	10000	2354,02	6420,03
	Video3-T3	10000	10000	6432,97	8810,99
Sum3		24514,2	30000	9418,78	21311,01
Total Sum (Sum1 + Sum2 + Sum3)		46214,85	58750,06	70756,02	
Faults		1	5	2	

Finally, Figure 21.20 shows a comparison with all the cases and steps that we have presented in this work: individual optimization, scenario optimization, and global optimization. As it has been mentioned, instead of comparing particular solutions for individual targets or scenarios, each case is represented with the average evaluation over all cases. So, values in Tables 21.4, 21.5 are compared with the last columns in Tables 21.1, 21.2, 21.3. Furthermore, the total sum of all scenarios is shown below as the aggregated summary. It can be checked that the main goal of the work is achieved: the more general solution (more cases considered in the design), the better is the average performance. Moreover, it is relevant the good performance of the maximum operator in this process.

To close this subsection, it is interesting to analyze the fitness convergence in populations for both cases, optimization over each trajectory and over the worst case of a set of trajectories. The next figure illustrates the convergence (maximum, minimum and average fitness) for the first scenario, considering individual optimization, optimization of worst-case trajectory and sum for each scenario, and global optimization for all scenarios. The convergence is slower for the minimax strategy, since now the function to optimize is more complex, as the

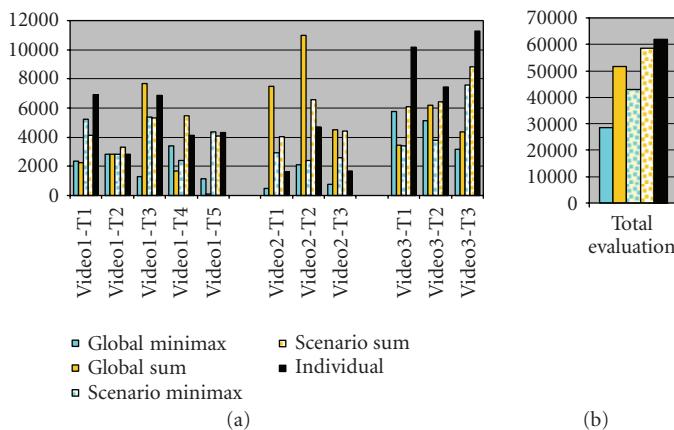


FIGURE 21.20. Comparison with all the cases and steps that we have presented in this work: individual optimization, scenario optimization and global optimization.

TABLE 21.4. Cross evaluation for parameters optimized for all scenarios with minimax.

evaluation scenario	designed scenario	All videos
	param-Videos	
Video 1	Video1-T1	2347,60
	Video1-T2	2820,85
	Video1-T3	1280,23
	Video1-T4	3416,05
	Video1-T5	1146,61
	Sum1	11011,34
Video 2	Video2-T1	494,70
	Video2-T2	2095,89
	Video2-T3	787,59
	Sum2	3378,18
Video 3	Video3-T1	5766,68
	Video3-T2	5136,36
	Video3-T3	3168,68
	Sum3	14071,72
Total Sum (Sum1 + Sum2 + Sum3)		28461,24

evaluation takes into account the set of situations in the three trajectories. As we have indicated, optimization with the sum function is faster but it is likely to fall in bad solutions, local minima with bad properties in other scenarios.

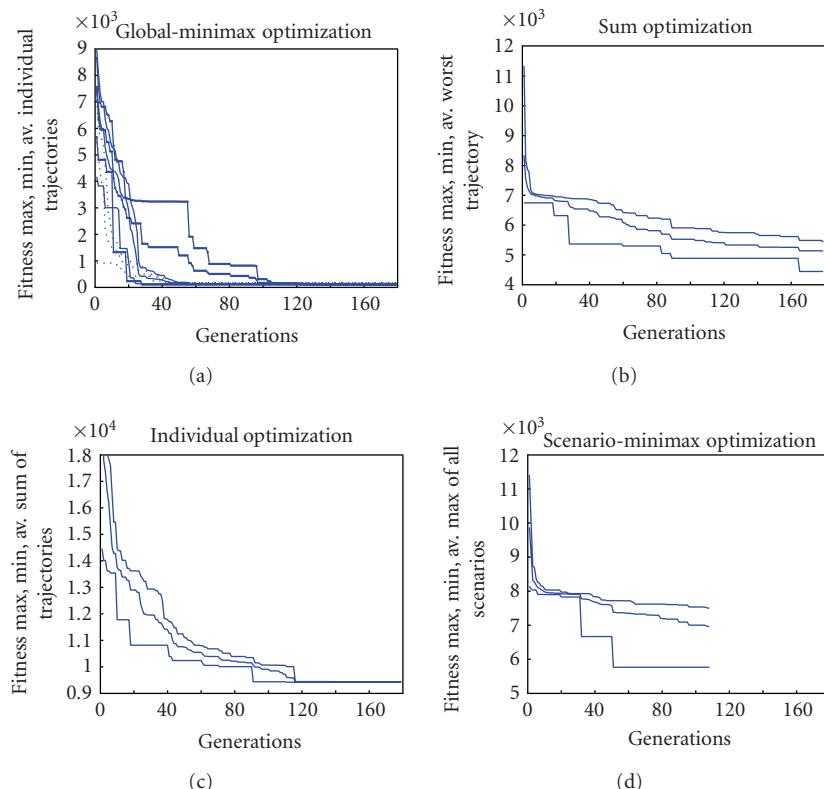


FIGURE 21.21. Fitness convergence for individual optimization, sum and worst-case for each scenario, and global worst case fitness functions.

21.6. Conclusions

In this chapter we have presented a novel process to adapt the performance of a tracking system based on the extraction of information from images captured by a camera and evolution strategies. We have proposed the application of ES to adapt the performance of a whole video tracking system with respect to real situations. The sets of parameters of the tracking system have been adjusted to obtain a good performance under very different situations (big aircraft interacting with small vans, occluded trajectories, different manoeuvres, etc.).

A significant improvement of the global vision system is achieved, in terms of accuracy and robustness. With this design methodology based on optimization, the inter-relation of parameters at different levels allows a coherent behavior under different situations. A generalization analysis has shown the capability to overcome the overadaptation when particular cases are considered and a continuous improvement when additional samples are aggregated in the training process, comparing two different operators: sum and worst-case aggregation.

TABLE 21.5. Cross evaluation for parameters optimized for all scenarios with sum.

	designed scenario	All videos
evaluation scenario		param-Videos
Video 1	Video1-T1	2243,12
	Video1-T2	2855,57
	Video1-T3	7683,49
	Video1-T4	1676,22
	Video1-T5	105,63
	Sum1	14564,03
<hr/>		
Video 2	Video2-T1	7506,24
	Video2-T2	10970,60
	Video2-T3	4523,21
	Sum2	23000,05
<hr/>		
Video 3	Video3-T1	3465,03
	Video3-T2	6181,07
	Video3-T3	4363,25
	Sum3	14009,35
<hr/>		
Total Sum (Sum1 + Sum2 + Sum3)		51573,43

Acknowledgments

This work has been funded by research projects CICYT TSI2005-07344, CICYT TEC2005-07186, CAM MADRINET S-0505/TIC/0255, and IMSERSO AUTOPIA.

Bibliography

- [1] B. R. Abidi, A. F. Koschan, S. Kang, M. Mitckes, and M. A. Abidi, "Automatic target acquisition and tracking with cooperative fixed and PTZ video cameras," in *Multisensor Surveillance Systems: The Fusion Perspective*, G. L. Foresti, C. Regazzoni, and P. Varshney, Eds., pp. 43–59, Kluwer Academic Publishers, Boston, Mass, USA, 2003.
- [2] *European Manual of Advanced Surface Movement and Control Systems (ASMGCS). Draft. Volume 1: Operational Requirements. Version 04*, 2000.
- [3] T. Bäck, *Evolutionary Algorithms in Theory and Practice*, Oxford University Press, New York, NY, USA, 1996.
- [4] T. Bäck, D. B. Fogel, and Z. Michalewicz, *Evolutionary Computation: Advanced Algorithms and Operators*, Institute of Physics, London, UK, 2000.
- [5] T. Bäck, D. B. Fogel, and Z. Michalewicz, *Evolutionary Computation: Basic Algorithms and Operators*, Institute of Physics, London, UK, 2000.
- [6] J. A. Besada, J. Portillo, J. García, J. M. Molina, Á. Varona, and G. Gonzalez, "Image-based automatic surveillance for airport surface," in *Proceedings of the 4th International Conference on Information Fusion (FUSION '01)*, pp. 11–18, Montreal, QC, Canada, August 2001.
- [7] J. Black, T. Ellis, and P. Rosin, "A novel method for video tracking performance evaluation," in *Proceedings of Joint IEEE International Workshop on Visual Surveillance and Performance Evaluation of Tracking and Surveillance (VS-PETS '03)*, pp. 125–132, Nice, France, October 2003.

- [8] S. Blackman and R. Popoli, *Design and Analysis of Modern Tracking Systems*, Artech House, Boston, Mass, USA, 1999.
- [9] R. Castaldo, C. C. Franck, and A. B. Smith, "Evaluation of FLIR/IR camera technology for airport surface surveillance," in *Enhanced and Synthetic Vision*, vol. 2736 of *Proceedings of SPIE*, pp. 64–74, Orlando, Fla, USA, April 1996.
- [10] I. Cohen and G. Medioni, "Detecting and tracking moving objects in video from an airborne observer," in *DARPA Image Understanding Workshop (IUW '98)*, pp. 217–222, Monterey, Calif, USA, November 1998.
- [11] F. Cupillard, F. Bremond, and M. Thonnat, "Tracking groups of people for video surveillance," in *Video-Based Surveillance Systems: Computer Vision and Distributed Processing*, pp. 89–100, Kluwer Academic Publishers, Dordrecht, The Netherlands, 2002.
- [12] A. P. Dempster, N. M. Laird, and D. B. Rubin, "Maximum likelihood from incomplete data via the EM algorithm," *Journal of the Royal Statistical Society. Series B*, vol. 39, no. 1, pp. 1–38, 1977.
- [13] D. Doermann and D. Mihalcik, "Tools and techniques for video performance evaluation," in *Proceedings of 15th International Conference on Pattern Recognition (ICPR '00)*, vol. 4, pp. 167–170, Barcelona, Spain, September 2000.
- [14] U.S. Department of Transportation. FAA. The Future Airport Surface Movement Safety, Guidance and Control Systems: a vision for Transition into the 21st Century. Washington, DC, USA, November 1993.
- [15] G. L. Foresti and C. Micheloni, "A robust feature tracker for active surveillance of outdoor scenes," *Electronic Letters on Computer Vision and Image Analysis*, vol. 1, no. 1, pp. 21–34, 2003.
- [16] D. A. Forsyth and J. Ponce, *Computer Vision: A Modern Approach*, Prentice-Hall, Upper Saddle River, NJ, USA, 2003.
- [17] L. M. Fuentes and S. A. Velastin, "Assessment of image processing as a means of improving personal security in public transport," in *Video-Based Surveillance Systems: Computer Vision and Distributed Processing*, pp. 159–166, Kluwer Academic Publishers, Dordrecht, The Netherlands, 2002.
- [18] J. García, J. A. Besada, J. M. Molina, J. I. Portillo, and G. de Miguel, "Fuzzy data association for image-based tracking in dense scenarios," in *Proceedings of the IEEE International Conference on Fuzzy Systems (FUZZ '02)*, vol. 2, pp. 902–907, Honolulu, Hawaii, USA, May 2002.
- [19] J. G. Herrero, J. A. B. Portas, A. B. de Jesús, J. M. M. López, G. de Miguel Vela, and J. R. C. Corredora, "Application of evolution strategies to the design of tracking filters with a large number of specifications," *EURASIP Journal on Applied Signal Processing*, vol. 2003, no. 8, pp. 766–779, 2003.
- [20] J. García, J. M. Molina, J. A. Besada, and J. I. Portillo, "A multitarget tracking video system based on fuzzy and neuro-fuzzy techniques," *EURASIP Journal on Applied Signal Processing*, vol. 2005, no. 14, pp. 2341–2358, 2005.
- [21] H. Gauvrit, J. P. Le Cadre, and C. Jauffret, "Formulation of multitarget tracking as an incomplete data problem," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 33, no. 4, pp. 1242–1257, 1997.
- [22] Manual of SMGCS. ICAO. Doc. 9476-AN/927.
- [23] Y. Kuno, "Detecting and tracking people in complex scenes," in *Video-Based Surveillance Systems: Computer Vision and Distributed Processing*, Kluwer Academic Publishers, Dordrecht, The Netherlands, 2002.
- [24] T. M. Mitchell, "Generalization as search," *Artificial Intelligence*, vol. 18, no. 2, pp. 203–226, 1982.
- [25] K. Ohkura, Y. Matsumura, and K. Ueda, "Robust evolution strategies," *Applied Intelligence*, vol. 15, no. 3, pp. 153–169, 2001.
- [26] J. H. Piater and J. L. Crowley, "Multi-modal tracking of interacting targets using Gaussian approximation," in *Proceedings of 2nd IEEE Workshop on Performance Evaluation of Tracking and Surveillance (PETS '01)*, pp. 141–147, Kauai, Hawaii, USA, December 2001.
- [27] D. Pokrajac and L. J. Latecki, "Spatiotemporal blocks-based moving objects identification and tracking," in *Proceedings of Joint IEEE International Workshop on Visual Surveillance and Performance Evaluation of Tracking and Surveillance (VS-PETS '03)*, Nice, France, October 2003.
- [28] I. Rechenberg, *Evolutionsstrategie'94*, Frommann-Holzboog, Stuttgart, Germany, 1994.

- [29] P. L. Rosin and E. Ioannidis, "Evaluation of global image thresholding for change detection," *Pattern Recognition Letters*, vol. 24, no. 14, pp. 2345–2356, 2003.
- [30] M. Sonka, V. Hlavac, and R. Boyle, *Image Processing, Analysis, and Machine Vision*, Brooks/Cole, Belmont, Calif, USA, 1999.
- [31] C. E. Schwab and D. P. Rost, "Airport surface detection equipment," *Proceedings of the IEEE*, vol. 73, no. 2, pp. 290–300, 1985.
- [32] H.-P. Paul Schwefel, *Evolution and Optimum Seeking: The Sixth Generation*, John Wiley & Sons, New York, NY, USA, 1995.
- [33] B. W. Wah, "Generalization and generalizability measures," *IEEE Transactions on Knowledge and Data Engineering*, vol. 11, no. 1, pp. 175–186, 1999.
- [34] D. H. Wolpert and W. G. Macready, "No free lunch theorems for optimization," *IEEE Transactions on Evolutionary Computation*, vol. 1, no. 1, pp. 67–82, 1997.

Jesús García: Departamento de Informática, Universidad Carlos III de Madrid,
Avda. Universidad Carlos III, Colmenarejo, Madrid 28270, Spain

Email: jgherrer@inf.uc3m.es

Óscar Pérez: Departamento de Informática, Universidad Carlos III de Madrid,
Avda. Universidad Carlos III, Colmenarejo, Madrid 28270, Spain

Email: opconcha@inf.uc3m.es

Antonio Berlanga: Departamento de Informática, Universidad Carlos III de Madrid,
Avda. Universidad Carlos III, Colmenarejo, Madrid 28270, Spain

Email: aberlan@ia.uc3m.es

José M. Molina: Departamento de Informática, Universidad Carlos III de Madrid,
Avda. Universidad Carlos III, Colmenarejo, Madrid 28270, Spain

Email: molina@ia.uc3m.es