1 Genetic and Evolutionary Computation for Image Processing and Analysis

Stefano Cagnoni, Evelyne Lutton, and Gustavo Olague

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.

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.

[1] On 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]).

[2] See [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.

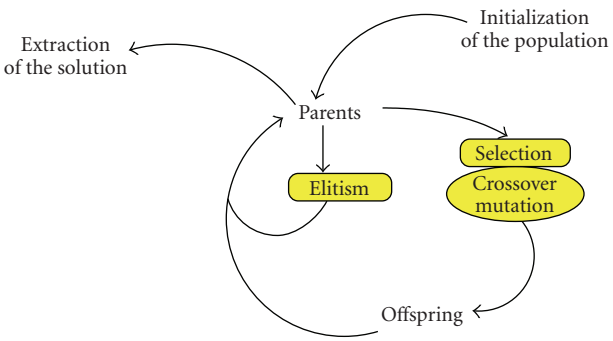

Figure 1.1. Organigram of a simple evolutionary algorithm.

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:

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

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.).

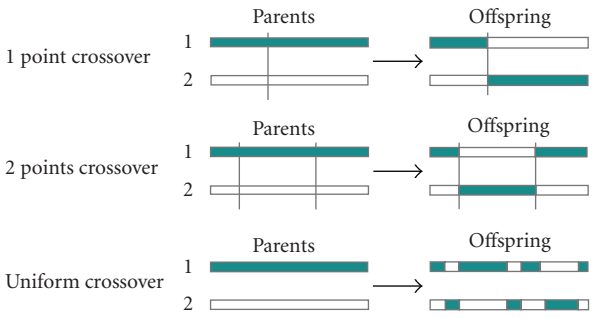

FIGURE 1.2. Binary crossover.

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

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

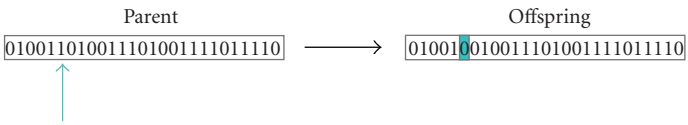

FIGURE 1.3. Binary mutation.

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

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

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:

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

[4]It 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].

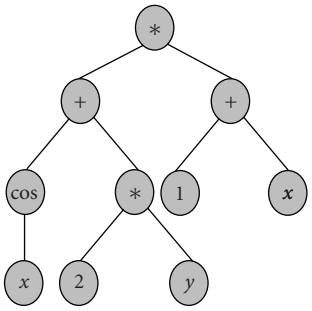

FIGURE 1.4. Example of a tree representation of the function $((\cos(x) + 2 * y) * (1 + x))$.

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).

[5] A 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 auditive). 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 algoritms 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 *Proccedings 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," IlliGAL 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. Puente, "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. Marcelin, and C. Schmelding, "Optimal damping of viscoelastic constrained beams or plates by use of a genetic algotithm," 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 satisfation 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