IMPROVING THE PERFORMANCE OF EVOLUTIONARY ALGORITHMS IN IMAGING OPTIMIZATION

by

Igor V. Maslov

A dissertation submitted to the Graduate Faculty in Computer Science in partial fulfillment of the requirements for the degree of Doctor of Philosophy,
The City University of New York

2008

UMI Number: 3296911

ProQuest Information and Learning Company
300 North Zeeb Road
P.O. Box 1346
Ann Arbor, MI 48106-1346

This manuscript has been read and accepted for the Graduate Faculty in Computer Science in satisfaction of the dissertation requirement for the degree of Doctor of Philosophy.

Professor Izidor Gertner

June 18, 2007

Date — Chair of Examining Committee

Professor Theodore Brown

July 11, 2007

Date — Executive Officer

Professor Ioannis Stamos

Professor Zhigang Zhu

Professor Ilya Muchnik

Supervisory Committee

THE CITY UNIVERSITY OF NEW YORK

Abstract

IMPROVING THE PERFORMANCE OF EVOLUTIONARY ALGORITHMS IN IMAGING OPTIMIZATION

By

Igor V. Maslov

Advisor: Professor Izidor Gertner

This research applies Evolutionary algorithms (EAs) to the task of finding a proper mapping between geometrically distorted images, which arises in applications like image registration and object recognition. The task is formulated as an imaging optimization problem of minimizing the difference between the images. The objective of the research is to improve the computational performance of EAs in imaging optimization, by developing a fairly general approach based on a broader hybridization of EAs with the following techniques.

- Augmenting EAs with local optimization technique, in the form of a two-phase (random/direct) cyclic search procedure reducing the excessive computational cost of local search.
- Utilizing a frontal algorithm of forming new population assuring its diversity, and restoring the fair and effective usage of the search space disrupted by evolutionary operators.

- Utilizing Image local response which directly extracts the main shape features; reduces the computational cost of fitness evaluation; and provides an efficient image model for adaptive local search, reduction of parameter space, and multi-sensor image fusion.
- Introducing an advanced image model which reduces the amount of processed information, and includes the following steps: computing Image response, building response histogram, decomposing image into sections, decomposing sections into quadtrees, and defining the main shape feature, a hull.
- Representing the sought image mapping as a piece-wise affine transformation allowing for significant mutual distortion of the compared images, so that different image sections have their respective local affine transformations.
- Organizing image sections in a tree-structure which is processed in a hierarchical top-to-bottom manner, with local transformations of parental sections serving as initial approximations for local transformations of the offspring.
- Utilizing multiple aligned populations aimed at increasing the coherence and robustness of the mapping during simultaneous processing of multiple views.
- Utilizing multi-objective optimization, where different fitness functions are processed at the different computational stages, thus increasing the confidence of the search.
- Implementing optimization search as two consecutive passes. During the first pass, global optimization seeks for a proper mapping of the image hull. During the second pass, the hull transformation is used as an initial solution for the final piece-wise optimization.

ACKNOWLEDGEMENTS

I would like to express my sincere gratitude to my Academic Advisor, Professor Izidor Gertner, for his continued guidance and assistance over the entire course of my work on this dissertation.

I would like to thank the committee members Professor Ioannis Stamos, Professor Zhigang Zhu, and Professor Ilya Muchnik. Their assistance and guidance was a great help for me, and without this help I would not have been able to complete my dissertation.

I would like to thank Professor Theodore Brown for his help and support while studying at the Graduate Center. Moreover, I would like to add a great thanks to Mr. Joseph Driscoll. I would also like to thank Mr. Juan Pagan.

I would like to add special thanks to Dr. Lawrence Nii Nartey for his continued help and support over my entire graduate studies.

I cannot forget my family who provided me with moral support and encouragement. I would like to give my special warm thanks to Ms. Meiri Kanamaru for being patient, helpful, and supportive during the last years of my studies.

Table of Contents

List of Figures

List of Tables

Chapter 1: Introduction

1.1. The Main Objectives of Research and Organization of the Thesis

Computerized processing of visual information obtained in traditional optical, thermal, or electronic form is an essential part of many scientific, technical, industrial, military, and other real-world applications. Large group of computerized visual applications including remote sensing and photogrammetry, industrial control, medical imaging, computer and robotic vision, multi-sensor fusion, and security and defense have to deal, in one way or another, with the problem of automated or semi-automated mapping between different images of the same object or a scene, when images might distorted by some geometrical transformation(s). A plethora of computer-aided models, methods, and algorithms have been developed ranging from rather general approaches like correlation, mutual information, and artificial neural networks, to numerous ad-hoc methods and techniques designed for a particular type of imagery, or specific transformation. However, continually growing complexity of imaging problems and necessity to deal with continuously changing dynamic and uncertain information put higher demands on the existing systems of processing visual information, and expose their limitations and frequent inefficiency, in terms of both, the processing time, and the ability to solve large-scale problems.

Facing the growing challenges of the real-world imaging applications, researchers turn their attention to new methods that are capable of autonomous adjustment and self-adaptation to the volatile inputs and tasks. One of the powerful and versatile among such

methods is a family of Evolutionary algorithms (EAs) – heuristic-based global search and optimization methods that have found their way into almost every area of real-world applications. Make a note that throughout this work, the term Evolutionary algorithm(s) is used in its broad sense, as a reference to the multiplicity of methods and algorithms, originally based on the principles of natural evolution. The algorithms have three principal advantages over other computational methods:

- EAs use inherently parallel processing of information;
- EAs are efficient in conducting global search in a large multi-dimensional parameter space;
- EAs are highly tolerant to non-linearity, non-continuity, and irregularity of the objective function.

In order to use Evolutionary algorithms, the original problem of image mapping has to be reformulated as an optimization problem. The term *imaging optimization* in this thesis refers to the broad problem of the mapping (i.e., geometrical transformation) of different and possibly distorted images of the same object or scene onto a common reference system, formulated as an optimization problem of minimizing the difference between the images subject to the mapping. Such problem is an essential part of many practical imaging applications, e.g., image registration, object and target recognition, pattern recognition, scene interpretation, multi-sensor fusion, and content-based image retrieval.

Analysis of the state-of-the-art in the Evolutionary algorithms, on the one hand, and its comparison with the range of the practical imaging optimization problems solved with

EAs, on the other hand, shows significant gap between the two. While the current status in the EAs theory and methodology includes a broad variety of advanced models and techniques, the majority of the practical imaging applications use the basic, classical models of the main representatives from the EAs family. The classical EAs have proven inefficient in many practical applications, including some of the problems related to image processing. However, the implementation of advanced methods that go beyond the classical models requires an in-depth understanding of the morphological development of Evolutionary algorithms, as well as a fairly comprehensive analysis of their recent trends and advances.

Challenges in solving complex real-world imaging optimization problems, and the shortcomings of the EAs models currently used to solve these problems, suggest the overall objective of this research: it attempts to analyze the potential use of modern advances in Evolutionary computations for solving practical problems encountered in imaging optimization. In particular, the research attempts to achieve the following objectives:

- develop and analyze a range of models, algorithms, and techniques implementing advanced concepts of Evolutionary algorithms, which can improve their computational performance in imaging optimization;
- design a fairly general approach based on a broader hybridization of EAs with a range of other advanced models, algorithms, and techniques, for solving a larger spectrum of imaging optimization problems.

Although the strong emphasis of this research is on the practical aspects of the EAs implementation, including specific and detailed models, algorithms, and techniques, the practical aspects are discussed from the perspective of the development of a unified platform for solving a variety of imaging optimization problems, rather than from the standpoint of the details of a particular application, e.g., image registration, or object recognition. This approach emphasizes the generality of the developed models, algorithms, and techniques, across different applications of imaging optimization.

With respect to the stated objectives of the research, the organization of this work is as follows. The rest of the introductory Chapter 1 gives the statement of the imaging optimization problem, together with a brief comparative overview of three candidate methods for solving the stated problem – Simulated annealing, Tabu search, and Evolutionary algorithms.

Chapter 2 provides a brief overview of computational methods commonly used in various imaging optimization applications, such as image registration, object recognition, and content-based image retrieval. In particular, correlation techniques, mutual information, cross entropy, iterative closest point algorithm, and artificial neural networks are discussed.

Chapter 3 gives a detailed insight into morphological development and the state-of-the-art in Evolutionary algorithms, including both theoretical concepts and their practical

implementations, in a variety of applications. In particular, an overview of EAs in solving real-world imaging optimization problems is given.

Chapter 4 starts with computational experiments demonstrating unsatisfactory performance characteristics of the classical version of Evolutionary (in particular, Genetic) algorithm. The concept of hybridization of EAs in a broader sense, with other computational techniques is discussed in detail. In particular, various local optimization techniques are analyzed; parameter space bookkeeping during mutation is introduced; and providing the algorithm with the essential information about images, in the form of partial image evaluation, is discussed.

Chapter 5 introduces a fundamental concept of Image local response, and discusses its use for improving the performance of EAs, including selective fitness evaluation, adaptive local search, adaptive selection and crossover, and multi-sensor fusion.

Chapter 6 proposes a general platform for solving a broader spectrum of imaging optimization problems, which employs multiple populations, multiple optimization objectives, and piece-wise affine transformation of the distorted images subject to the mapping. The algorithm is illustrated on three sample computational problems: human body registration, 3-D object recognition in the 2-D space, and elastic registration of medical images.

Chapter 7 concludes this work with the summary of the findings and brief discussion of the potential directions of further research.

1.2. Statement of Imaging Optimization Problem

A few problems encountered in imaging applications can be formulated in a unified manner, as a global optimization problem. These problems include image registration, object recognition, multi-sensor fusion, and content-based image retrieval. Image registration is commonly used in such areas as remote sensing, industrial control, and medical imaging applications. The problem is stated as follows. Given two images, a reference image Img_0, and a test image Img_1 subject to registration, geometrical transformation has to be found that correctly maps the points of Img_1 into corresponding points of Img_0. Object recognition is a common task in such areas as computer and robotic vision and automatic target recognition. It can be stated as the problem of finding a particular object having a signature (i.e., image) Img_1, in a scene Img_0 that might contain many different objects. In multi-sensor fusion, images obtained from different sensors have to be mapped into a common reference system. In content-based image search and retrieval, a task arises of finding the original scene Img_0 in a large image database, such that Img_0 contains a particular object Img_1, called a query image.

All aforementioned imaging problems essentially look for a proper transformation A that would provide a "good" match between the corresponding points of the images Img_1 and Img_0. If Img_1 and Img_0 are 2-dimensional grayscale images, then one simple and straightforward way to evaluate the quality of the match is to compute the difference

between the pixel values of the images, after the transformation A has been applied to one of them (e.g., Img_1). The minimum value of the difference would indicate a potential match between the images, with the corresponding optimum values of the parameters defining the transformation A.

Without loss of generality, as an example, one can consider an affine transformation A that includes image translation, rotation, and non-isotropic scaling. Then, the transformed vector $\boldsymbol{p}' = \{x',y'\}^{\mathrm{T}}$ of the coordinates $\boldsymbol{p} = \{x,y\}^{\mathrm{T}}$ of the pixel $P \in Img_1$ can be found as

$$\boldsymbol{p}' = A(\boldsymbol{p}) = SR\boldsymbol{p} + T, \tag{1.1}$$

where the matrices S, R, and T denote scaling, rotation, and translation, respectively [12].

For example, for a 2-dimensional image Formula (1.1) becomes

$$\begin{Bmatrix} x' \\ y' \end{Bmatrix} = \begin{vmatrix} SX & 0 \\ 0 & SY \end{vmatrix} * \begin{vmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{vmatrix} * \begin{Bmatrix} x \\ y \end{Bmatrix} + \begin{Bmatrix} DX \\ DY \end{Bmatrix}, \tag{1.2}$$

where SX and SY are the scaling factors along the x- and y-axes, θ is the rotation angle, and DX and DY are the translations along the x and y axes [12]. Make a note that the matrices R and T correspond to the rigid body transform, while the matrix S describes the local distortion of the image when the scaling factors are non-isotropic, i.e., $SX \neq SY$.

Transformation A in Equation (1.1) can also be written in a form of a combination of global transformation defining the location of the object in the scene (i.e., the rigid body transformation), and local transformation defining the deformation (i.e., the distortion) of the object, as follows:

$$A(\boldsymbol{p}) = LG_1\,\boldsymbol{p} + G_2, \tag{1.3}$$

where L is the local transformation caused by the matrix S of the non-isotropic scaling, G_1 is the part of the global transformation G due to the object rotation defined by the matrix R, and G_2 is the part of the global transformation G due to the translations along the x and y axis defined by the matrix T.

The problem of finding the correct transformation A in (1) can be formulated as the problem of finding a feasible vector of parameters $\boldsymbol{V}^* = \{DX, DY, \theta, SX, SY\}$ minimizing the difference F between the images Img_1 and Img_0, such that

$$A^*(\boldsymbol{V}^*) = \arg\min (F(\boldsymbol{V})), \text{ or } F(\boldsymbol{V}) > F^*(\boldsymbol{V}^*), \text{ for all } \boldsymbol{V} \neq \boldsymbol{V}^*. \tag{1.4}$$

One direct and frequently used way to measure the difference F between the images is to compute the squared difference of the gray values of the two images, over the squared area of their overlap, i.e.,

$$F = \frac{\sum_{\Omega} (g_1(X', Y') - g_0(X, Y))^2}{\Omega^2}, \tag{1.5}$$

where $g_1(x', y')$ and $g_0(x, y)$ are the gray values of the images Img_1 and Img_0, respectively, and Ω is the area of their overlap [12].

In imaging applications, the difference measure F defined in (1.5) is a nonlinear function that does not have a closed analytical form, and has to be computed numerically for every trial parameter vector $\boldsymbol{V}$. In the case when the reference image Img_0 might be a complex, cluttered, and noisy scene, and the object shape corresponding to the image Img_1 might be significantly distorted by the transformation A, the problem (1.4) - (1.5) turns into a

nonlinear, multimodal global optimization problem, where the correct global solution can be obscured by multiple deceptive local solutions. If there is no prior information available about the images, the search space of potential solutions becomes so large that the brute force approach becomes intractable. The classical optimization techniques fail due to the presence of multiple local minima, and in general case, a non-convex shape of the nonlinear function *F*. Stochastic optimization methods have proved to be efficient approach to solving this kind of problems [30]. Among stochastic methods, three representatives - Simulated annealing, Tabu search, and Evolutionary algorithms, – received a good deal of attention in practice and research.

1.3. Stochastic Global Optimization Methods in Imaging Optimization

This section discusses the algorithms and computational performance of three powerful stochastic methods of global optimization, Simulated annealing, Tabu search, and Evolutionary algorithms. All three methods have been developed to find the globally optimal solutions in problems that might contain multiple local minima. Optimization problems of this kind are significantly more difficult than linear problems that can be solved by the means of linear programming.

1.3.1. Simulated Annealing

Simulated annealing (SA) uses the global search strategy that helps algorithm climb out of a local minimum, and continue the search toward the global optimum solution [124]. The strategy draws on the observation of physical processes that occur in cooling solids, after they first have been heated to a liquid state. If the cooling schedule is slow enough,

it leads to the minimal energy state, and the formation of crystals. The algorithm starts at the chosen initial position $\boldsymbol{V}_0$ in the search space, with the initial temperature T_0. From the current position $\boldsymbol{V}_0$, the algorithm picks the next position $\boldsymbol{V}$ in the random direction with the step size $\boldsymbol{S}_V$, using the uniform probability distribution. The difference ΔF between the new, $F(\boldsymbol{V})$ and the current, $F(\boldsymbol{V}_0)$ values of the function F is computed and tested for the acceptance. If $\Delta F < 0$, i.e., the new position $\boldsymbol{V}$ is better than the current position $\boldsymbol{V}_0$, the algorithm accepts $\boldsymbol{V}$ and moves to the new position. If $\Delta F > 0$, i.e., the new position $\boldsymbol{V}$ is worse than the current position $\boldsymbol{V}_0$, the algorithm evaluates the probability of the acceptance of $\boldsymbol{V}$ using the Boltzmann probability distribution

$$p = \mathrm{e}^{(-\Delta F / kT)}, \tag{1.6}$$

where k is Boltzmann's constant, and T is the current temperature (initially, equals T_0) [124]. If $p > r$, where r is a number picked at random, with the uniform probability, from the interval (0, 1), then the new position is accepted. If $p < r$, then the position $\boldsymbol{V}$ is rejected, and the algorithm continues its search for a new position in the search space. When the algorithm falls into a local minimum point, it is the occasional acceptance of the position that is worse than the current local minimum, that makes the algorithm climb up the local hill and continue the search for a better (hopefully, the global) optimal solution. As the algorithm progresses, and the temperature decreases, the probability p is changing. The lower the temperature, i.e., the closer the system to its equilibrium state, the lower is the probability of climbing uphill.

The algorithm proceeds until no further movement is possible, i.e., the system has achieved the frozen equilibrium state, or until the number of the outer iterations has

exceeded the allowed maximum value O_{max}. The success of the algorithm significantly depends on the value of Boltzmann's constant k, as well as on the cooling schedule. The latter is defined by the following parameters:

- initial temperature T_0,
- maximum number of inner iterations I_{max} allowed at a given temperature,
- maximum number of position modifications M_{max} allowed at a given temperature,
- rate R_T at which the temperature decreases.

1.3.2. Tabu Search

Tabu search, also known as *adaptive memory search*, modifies an existing heuristic search by keeping a list of points in the search space that were most recently visited by the search algorithm [46]. These points then become *tabu* for the algorithm, i.e., these points are not allowed to be re-visited, as long as they are present on the list. This modification allows the search algorithm to eventually climb up out of the local minimum. Tabu search reportedly requires less computation than Simulated annealing, while providing similar, or even better results.

1.3.3. Evolutionary Algorithms

The family of Evolutionary algorithms belongs to the category of global search and optimization methods, and is inspired by the principles of natural evolution and genetics, in particular, by Neo-Darwinian theory of evolution [4, 66, 111]. The algorithms draw from observations of physical processes that occur in nature, and attempt to mimic them in the artificial computational environment. By and large, EAs have winning points over

traditional optimization techniques in solving nonlinear, non-convex, non-separable, discontinuous, non-differentiable, multimodal, noisy, and other computationally hard real-world problems, where traditional methods frequently fail or perform poorly.

Following the analogy with living organisms, the algorithms use the term *chromosome* to denote a single candidate solution to an optimization problem. Smaller blocks that constitute the solution are called after biological *genes*. In multidimensional optimization, a chromosome represents the vector of parameters that constitute a potential solution, while a gene is a single component of the vector. In particular, this work interprets the vector of the sought image transformation parameters $\boldsymbol{V}$ as a chromosome, whose genes are the individual components DX, DY, etc., of the vector $\boldsymbol{V}$. Unlike most traditional iterative optimization methods that work with a single potential solution, Evolutionary algorithms work with the population of solutions (i.e., chromosomes, or parameter vectors), concurrently evaluating and comparing the quality of all solutions in the population.

In natural evolution, the success of the living organism's adaptation to the environment is measured by the organism's *fitness*. In Evolutionary algorithms, fitness analogically serves as the measure of the quality of the potential solution, and typically corresponds to the objective function of the optimization problem, or some form of its surrogate. In particular, the image difference F defined in (1.5) can play the role of fitness that evaluates the quality of every individual chromosome.

Following the analogy with natural evolution, Evolutionary algorithms attempt to solve the optimization problem by searching for fitter chromosomes (i.e., parameter vectors) that have smaller (in the minimization problem) or larger (in the maximization problem) fitness values. In the particular problem of finding the parameter vector $\boldsymbol{V}^*$ of the transformation A^* minimizing the image difference F defined in (1.5), the classical elitist version of Evolutionary (in particular, Genetic) algorithm works as shown in Figure 1.1.

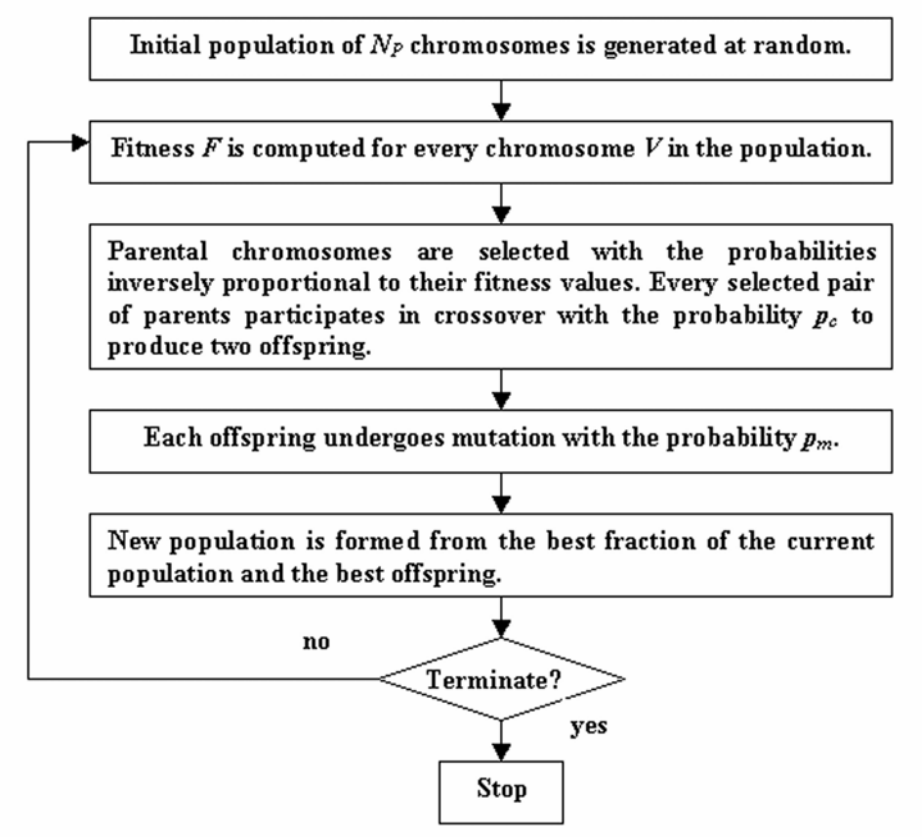

Figure 1.1: Search strategy of the classical elitist Evolutionary algorithm

The algorithm starts with the initial population P_I of chromosomes, or parameter vectors $\boldsymbol{V}_i$, where $i = 1,\ldots, N_P$, and N_P is the size of the population. Each of the components

(genes) DX_i, DY_i, etc., can take on its value from a generally large domain, which makes the task of building the population from all possible combinations of the components intractable. For example, for 256×256-pixel image with a possible range of the rotation angle $\theta = 0° - 360°$, at the step size of 1°, and a range of the scaling factors SX and SY between 1 and 5, at the step size of 0.1, the total number of possible combinations of the components is 256×256×360×40×40 = 37,748,736,000. It means that the exhaustive search over all possible parameter vectors would require the total population of 37,748,736,000 chromosomes. Since in reality, the size N_P of the population is limited, due to the limited computational resources, the value of each component forming the chromosome $\boldsymbol{V}$ is drawn independently, at random from the domain of all possible component values.

Every chromosome $\boldsymbol{V}$ specifies a particular position (e.g., the values of translations, rotation, and scaling factors) of the object Img_1, in relation to the scene Img_0. The quality of the chromosome is defined via the distance D from its current position in the search space to the optimum point indicating the exact match between the images Img_1 and Img_0 of the same object. The distance D evaluates the chromosome's fitness value, and is defined in (1.5) as the difference F between the two images. After the fitness value has been computed for every chromosome in the population P_I, parental chromosomes are selected to form the next generation, in such a way that the probability of the chromosome to become a parent is somewhat inversely proportional to its fitness value. The selection strategy, therefore, favors chromosomes with lower fitness values constituting smaller differences between the images.

1.3.4. Comparative Performance of Stochastic Optimization Methods

In order to evaluate the performance of the three aforementioned stochastic algorithms, computational experiments are conducted for two synthetic images modeling sensor readings, in accordance with the following Formula [12]:

$$g(x,y) = 100.0 + (-40x + 45y - 0.003xy + 0.02xx - 0.01yy - 20y \sin(x/18) + 35y \cos(y/29) - 35 \sin(x/4 - y/12) + 12x \cos(xy/100))/100. \quad (1.7)$$

The simulated sensor readings Img_0 and Img_1 shown in Figure 1.2 are two overlapping 256×256-pixel grayscale noisy images. The second (template) image Img_1 is displaced by $DX = 91$, $DY = 91$ from the center of the first (reference) image Img_0, and rotated by $\theta =$ 2.74889 radians (157.5°). Figure 1.3 shows the known exact solution, with the overlapping area for both images.

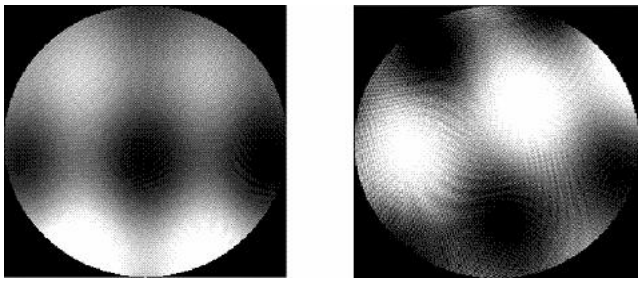

Figure 1.2: Synthetic sensor readings: reference (left) and template (right) images

The parameter vector $\boldsymbol{V}$ defining the problem includes translations DX and DY; and rotation angle θ between the images. The objective function F is defined as the squared difference between the pixel values, over the squared area of the overlap of the two images – see Formula (1.5).

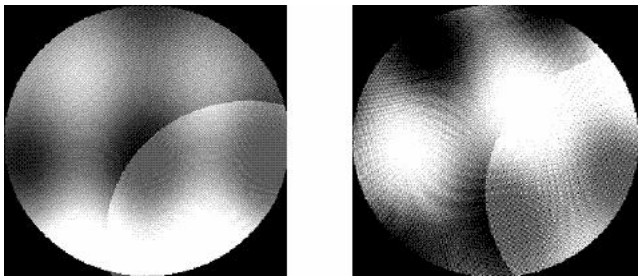

Figure 1.3: Overlapping areas for the exact solution: reference (left) and template (right) images

A weighted Euclidean distance δ between the approximate and exact solutions is used to measure the accuracy of the algorithm, i.e.

$$\delta = \sqrt{\left(\frac{\Delta(DX)}{DX}\right)^2 + \left(\frac{\Delta(DY)}{DY}\right)^2 + \left(\frac{\Delta(\theta)}{\theta}\right)^2}, \tag{1.8}$$

where $\Delta(DX)$, $\Delta(DY)$, and $\Delta(\theta)$ are the differences between the approximate and exact values of the components DX, DY, and θ.

In the Simulated annealing search, one of the six neighboring points (±1 pixel in the x direction, ±1 pixel in the y direction, and ±1° rotation θ) is chosen at random from the current position in the search space. The difference between the values of fitness function at the current and new points is computed; it is used to evaluate the probability that the new position will be accepted, at the current value of the system temperature. The probability of the acceptance is given by the Boltzmann distribution. The initial value of the temperature T is set at 1.1. The search continues at the same temperature T for either 5120 iterations, or until 512 positions have been tested, at which time the value of T is decreased by 5%. The temperature gradually decreases until no transitions are possible,

so the system remains frozen. Since this condition occurs only when ΔF is positive for all neighboring points, the attained position in the search space must be either a local, or the global minimum.

The implementation of the Tabu search starts with all parameters set to zero, and relies on a "greedy" heuristic, in order to find the next position (i.e., the parameter vector) in the search space. The search can move one pixel in the *x* direction, one pixel in the *y* direction, or rotate by one degree. The algorithm uses the image difference function *F* to evaluate the quality of the match of the two images, for each of six options (positive and negative values of *DX*, *DY*, and the value of θ). The search moves on to the next position, in the direction of the decrease of the value of the function *F*. After a position in the search space has been visited (i.e., the fitness value at this position has been evaluated), it is placed on the tabu list. The particular implementation of the Tabu search in these experiments uses an infinite tabu list.

Those positions in the search space that have been placed on the tabu list are excluded from the future evaluation, by setting the corresponding function value to a very large number. As each position in the search space is visited, the corresponding value of the function *F* is compared with the smallest value from the list of the parameter vectors that have already been visited, up to date. If the current value of *F* is smaller, the parameter vector now becomes the best candidate for the optimal solution. Since no well-defined termination criteria for Tabu search exist, and the only way to guarantee that the global minimum for the objective function has been found, is through an exhaustive search of

the entire search space, a preset maximum number of iterations 400 are used in these experiments.

The elitist reproduction scheme of the classical Evolutionary algorithm is used in these experiments. Under the elitist reproduction scheme, a certain number of chromosomes (20% here) with the best fitness values form the population of $P = 133$ are directly copied into the next generation, without undergoing crossover or mutation. The majority (77% here) of the rest of the new generation is formed by performing crossover between the chromosomes of the current population selected into the mating pool as follows. Each chromosome is assigned the probability of being chosen for the mating pool, based on the corresponding value of the fitness function F. Chromosomes with better values are more likely to be chosen. The mating pool is constructed by choosing chromosomes from the current population, according to the roulette wheel selection.

The next generation of chromosomes is formed by mixing (by means of the single-point crossover operation) the components of two parental chromosomes picked at random from the mating pool. The crossover probability is set here to $p_c = 1$, i.e., every selected parental pair is subject to crossover. The chromosome is split for crossover at a randomly chosen breakpoint, with all genes having equal probability of being chosen as the breakpoint. A small amount of mutation (3% here) exists in the system, where one parental gene is picked at random, and replaced with a random value from the corresponding parameter domain.

The results of the Simulated annealing run are presented in Table 1.1. The search starts from the center of the reference image. The algorithm eventually falls into a local minimum, and cannot climb out for the next 10 outer iterations. The search is terminated at the point $DX = -56$, $DY = 14$, $\theta = -0.27925$, with the relative error $\delta = 0.12426$ – see Table 1.1.

Outer iteration	***DX***	***DY***	$\boldsymbol{\theta}$	**Error** $\boldsymbol{\delta}$
0	-26	28	-0.08727	0.2625
20	-24	20	-0.48869	0.22052
40	-24	20	-0.48869	0.22052
60	-60	19	-0.33161	0.129
80	-57	18	-0.31416	0.12573
100	-56	14	-0.27925	0.12426
Exact solution	91	91	2.74889	0

Table 1.1: Results of the Simulated annealing run

Tabu search also starts from the center of the reference image. Since the algorithm does not have a well-defined terminating point, it is run until the error and parameter vector remained unchanged over 100 consecutive iterations. As one can see from the data shown in Table 1.2, the algorithm finds a local minimum point after 96 iterations, and cannot climb out from that point for the next 100 iterations, after which the search is terminated. The final solution $DX = -6$, $DY = 14$, $\theta = 6.00393$, with the error $\delta = 0.12426$, is located substantially away from the exact solution $DX = 91$, $DY = 91$, $\theta = 2.74889$.

Iteration	*DX*	*DY*	θ	Error δ
0	-1	0	0	0.33649
50	-51	0	0	0.17718
100	-56	14	6.00393	0.12426
150	-56	14	6.00393	0.12426
196	-56	14	6.00393	0.12426
Exact solution	91	91	2.74889	0

Table 1.2: Results of the Tabu search run

The elitist Evolutionary algorithm is terminated after 400 iterations; it shows a comparatively fast convergence to the global optimal solution – see Figure 1.4. The algorithm nearly reaches the global minimum point at iteration 54, with the parameter vector $\boldsymbol{V}^* = \{89, 84, 2.78081\}$. The minimum fitness value at that point is $F^* = 0.062$, and the weighted Euclidean distance from the exact solution is $\delta = 0.086$. During the evolutionary search, the process first reaches a local minimum point at iteration 9, with the parameters $DX = -34$, $DY = 73$, $\theta = 2.78081$, and the minimum fitness value $F = 0.14062$. It stays at approximately same level until iteration 45. The detailed analysis of the changes in the chromosome pool between the iterations helps explain why the stall occurs. Figure 1.5 shows the distribution of the fitness values in the chromosome pool across several iterations. Once a locally optimum parameter vector finds its way into the replication pool, it tends to propagate through the entire pool, as the result of a biased selection and multiple crossover operations. Only mutations performed on the reserved

fraction of the chromosome pool help maintain the diversity of the population, and prevent the search from being trapped at a local minimum point.

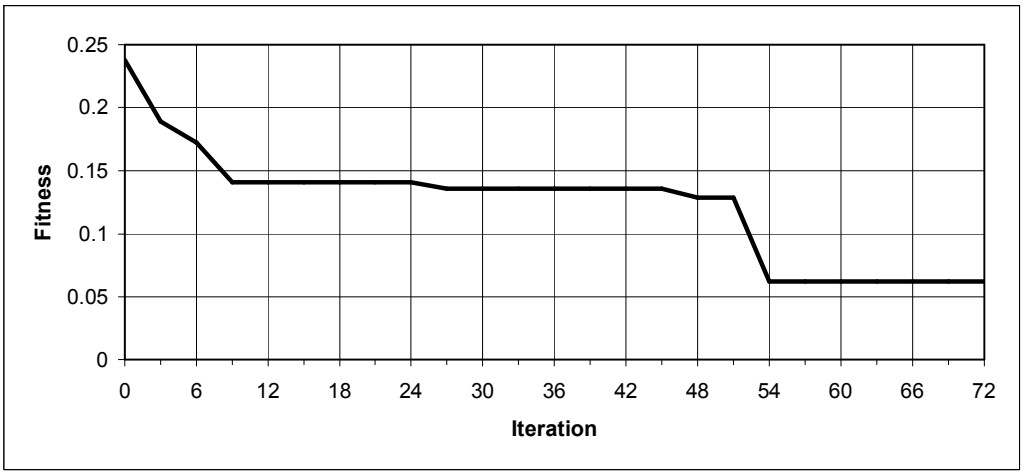

Figure 1.4: Performance of the elitist Evolutionary algorithm

1.3.5. Conclusions on Comparative Performance Analysis

The comparative analysis of the computational performance of Simulated annealing, Tabu search, and classical Evolutionary algorithm allows to make the following conclusions.

Both, Tabu search and Simulated annealing can be considered most efficient in finding the optimal position in the search space when the starting point is located fairly close to that position. Both algorithms perform poorly when the search starts at a relatively distant point from the global optimum; they tend to trap themselves in one of the local optimal points. Starting from the image center, the Simulated annealing run stalls at a local minimum point $\boldsymbol{V} = \{-56, 14, -0.27925\}$, and the Tabu search stalls at a point $\boldsymbol{V} = \{-6, 14,$

6.00393}. Both solutions are substantially distant from the global optimum $\boldsymbol{V}^* = \{91, 91, 2.74889\}$.

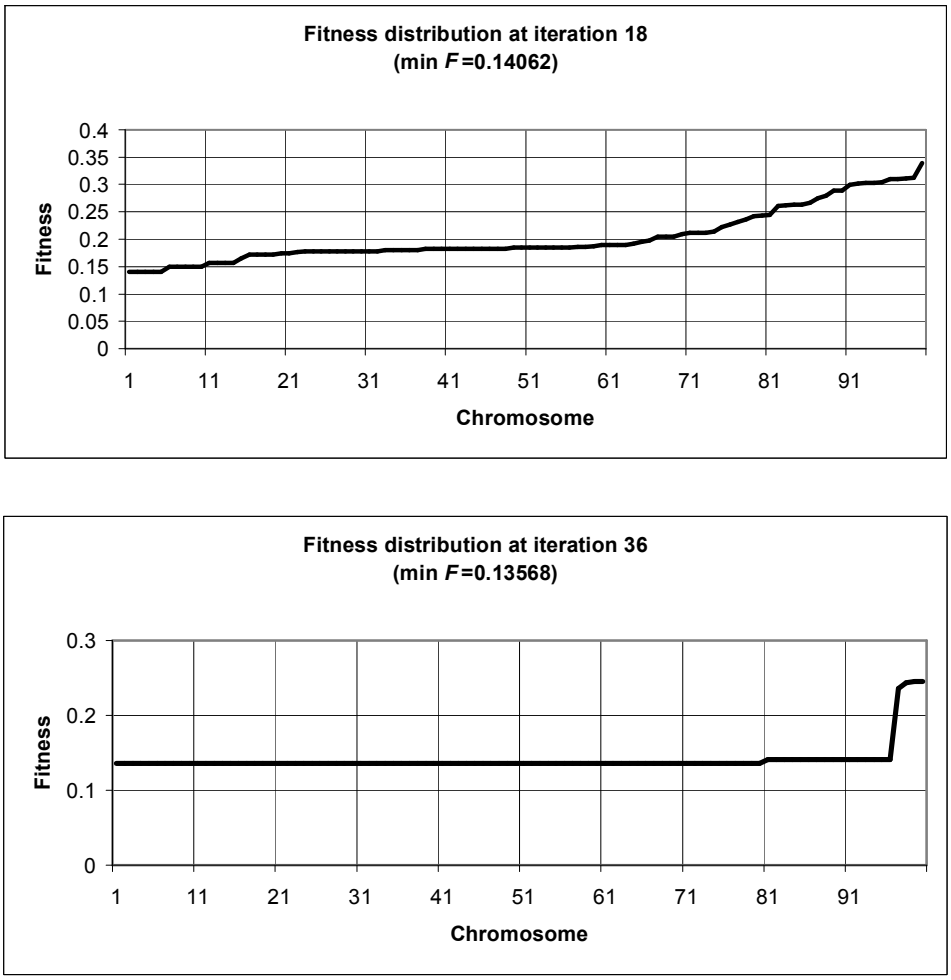

Figure 1.5: Fitness distribution in the chromosome pool over iterations

The classical elitist Evolutionary algorithm shows a fairly high performance in the global search; it reached a nearly global optimum $\boldsymbol{V} = \{89, 84, 2.78081\}$ after 54 iterations. One of the strong points of the algorithm is that it covers the entire search space at once, randomly (or heuristically) scattering and mixing the probe points. In general, however,

due to the inherent discrete character, the algorithm cannot reach the exact optimal solution while coming fairly close to it.

1.4. Summary

A fairly large group of visual applications relies on solving the problem of automated or semi-automated mapping between different images of the same object or a scene, when images might be distorted by some geometrical transformation(s). Imaging optimization formulates the image mapping problem as a global optimization problem of minimizing the difference between the images subject to the mapping, with an image difference playing a role of the objective function. The classical optimization techniques applied to imaging optimization often fail due to a nonlinear, multimodal, and non-convex character of the objective function, while stochastic optimization methods provide a powerful alternative in solving this type of problems. Comparative test runs of three powerful stochastic methods - Simulated annealing, Tabu search, and the classical elitist Evolutionary algorithm, - on a simple imaging optimization problem, show the superior performance of Evolutionary algorithm.

In many visual applications involving imaging optimization, the time required to obtain a solution is often a critical matter. Although the classical Evolutionary algorithm allows one to find the global optimum point in the parameter space, the computational time can be substantially large. Investigating the means of hybridization EAs with other advanced computational models, algorithms, and techniques that can potentially improve the EAs performance in imaging optimization, is the main objective of this research.

Chapter 2: Overview of Common Methods of Image Registration, Object Recognition, and Content-Based Image Retrieval

2.1. Image Registration

Image registration is an essential part of many image processing and computer vision applications, including medical imaging, industrial control, remote sensing, photogrammetry, security and defense, arts, and many others. Image registration integrates images obtained from different views of the same scene or the same object. All images have to be placed into the same reference framework, i.e., properly aligned. The registration procedure can be manual or semi-automated; in both cases, it includes intervention of a human operator. A certain number of salient points (called *control points*) identified by the operator are selected, in order to place the images into correspondence with each other. This type of registration is called landmark-based, or control-point-based.

The obvious drawbacks of the landmark-based registration techniques are the required human participation, and often a limited accuracy of registration related to human factors. From the point of view of automation, more interesting is an alternative approach based on extracting all relevant registration information directly from visual image content. Content-based registration involves either processing the raw pixel data, or processing the information obtained from a dual image representation in the feature space.

2.1.1. Traditional Two-Dimensional Registration

A number of approaches have been developed for 2-dimensional (2-D) registration of images subject to similarity transformation defined by translation, rotation, and isotropic scaling as follows:

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = S \times \begin{bmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} DX \\ DY \end{bmatrix}, \tag{2.1}$$

where the point $\boldsymbol{p}'(x',y')$ in the reference image corresponds to the point $\boldsymbol{p}(x,y)$ in the test, or template image, S is the scaling factor, θ is the rotation angle, and DX and DY are the translations of the test image, in relation to the reference image [12]. Commonly used methods of 2-D registration of images under the similarity transformation include cross correlation of the images in the spatial domain [82, 127, 134, 162], Fourier-based correlation [15, 23, 81, 92, 96, 147, 172], mutual information [20, 62, 69, 100, 119, 122, 152, 166, 167, 168, 173], and minimization of cross entropy [14].

In the conventional cross correlation between images, the template image Img_T is compared against the reference image Img_R in such a way that the image feature F in the template image has to be aligned with the corresponding image feature F in the reference image. The search window W scans over the image Img_T, in order to compute normalized correlation function $c(DX,DY)$ between the images for every position $O(DX,DY)$ of the window W [134]. The maximum value of the function $c(DX,DY)$ corresponds to the position of the search window at which the feature F is properly aligned, i.e., the template image Img_T is correctly registered with the reference image Img_R.

When images have large dimensions, finding cross correlation for different values of parameters defining image transformation becomes computationally expensive. Fourier-based correlation carried out in the frequency domain utilizes the high computational efficiency of the Fast Fourier Transform (FFT), and can significantly speed up (by the order of magnitude) computing the correlation function [147]. Another advantage of Fourier-based correlation is its higher robustness to varying noise and illumination, in comparison with the classical cross correlation.

Fourier-based correlation uses Fourier shift theorem and the fact that the peak of the cross power spectrum of two images is located at the point of their correct alignment [118]. The cross correlation of the images in the spatial domain turns into multiplication of their Fourier transforms in the frequency domain. If $F_R(u,v) = \Phi(g_R(i,j))$ is the Fourier transform of the image Img_R, $F_T(u,v) = \Phi(g_T(i,j))$ is the Fourier transform of the image Img_T, and $F_c(u,v) = \Phi(c(DX,DY))$ is the Fourier transform of the correlation function $c(DX,DY)$, then the following relationship holds:

$$F_c(u,v) = F_R(u,v) \times F_T^*(u,v) , \tag{2.2}$$

where $F_T^*(u,v)$ is the complex conjugate of the transform $F_T(u,v)$, and u and v are the variables in the frequency domain [118]. The original correlation function can be obtained [118] by taking the inverse Fourier transform Φ^{-1}, i.e.

$$c(DX,DY) = \Phi^{-1}(F_c(u,v)) = \Phi^{-1}(F_R(u,v) \times F_T^*(u,v)) . \tag{2.3}$$

When images are subject to rotation and scaling, in addition to translation, usually an exhaustive search is performed in the space of all possible parameter values, in order to

find the maximum value of the correlation function corresponding to the correct mapping of the images. For example, the first correlation peak can be found by varying the translation parameters, and the second correlation peak can then be found by varying the rotation angle. The accuracy of the correlation results degrades with the increasing values of the rotation angle and scaling factor.

The use of the conventional cross correlation and Fourier-based correlation as registration techniques is limited to the cases where:

- images are subject to similarity transformation that includes translation, rotation, and isotropic scaling, and excludes a noticeable distortion of the images;
- space of all possible geometric transformations is relatively small, which allows for conducting an exhaustive search for the proper values of transformation parameters, with the acceptable level of accuracy.

In many real-world applications, images are often subject to distortion, and registration has to be performed for some type of transformation, i.e., affine or perspective. Cross correlation fails to align images when they undergo distortion, in addition to the affine transformation. In order to demonstrate the behavior of cross correlation for the original, undistorted and distorted images, the computational experiments are conducted on the test images shown in Figure 2.1 [76].

The reference image Img_R (Figure 2.1, left) is a 2-D grayscale 256×256-pixel image, with 256 possible gray values ranging from 0 to 255, where 0 corresponds to black, and 255 to

white. The undistorted template image Img_T (Figure 2.1, right) is a 2-D grayscale 100×100-pixel image obtained by cropping the central portion (i.e., the region of interest) from the reference image. The center of the template image has coordinates $x = 130$, $y = 130$, i.e., the image Img_T is subject to the translation defined by the parameters $DX = DY = 130$. The correct location of the template image is shown with a black colored box in the reference image.

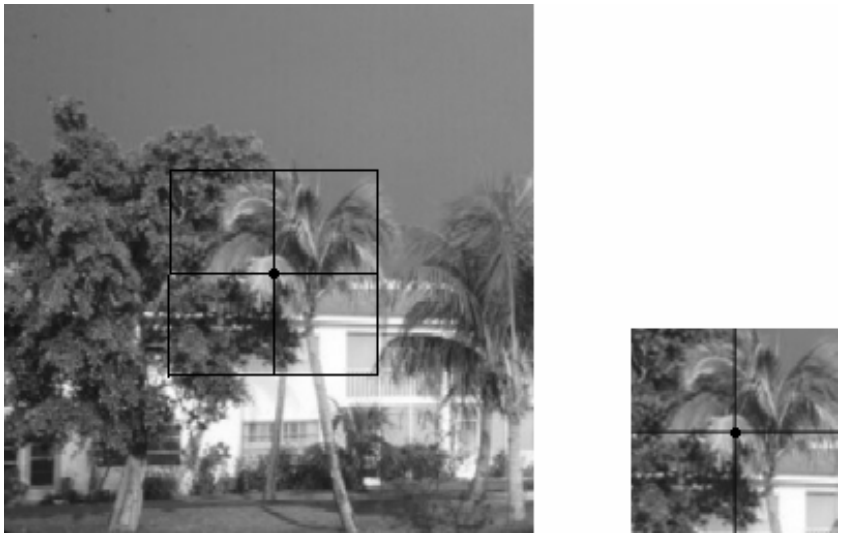

Figure 2.1: Reference image (left) and undistorted template (right) of a palm tree

The results of the cross correlation between the images Img_R and Img_T are shown in Figures 2.2 and 2.3. The 3-D plot of the correlation function $c(DX,DY)$ in Figure 2.2 has a peak corresponding to the correct mapping between the images. The left image in Figure 2.3 is a 2-D image of the correlation function, and the right image is the corresponding binary version of the correlation function obtained after applying the cut-off threshold of $c(DX,DY) = 0.95$. Location of the test image in the reference image is shown with a colored box. As one can see, the cross correlation finds the correct parameter values *DX*

= *DY* = 130, corresponding to the maximum value of the correlation function *c*(*DX*,*DY*) = 1.0.

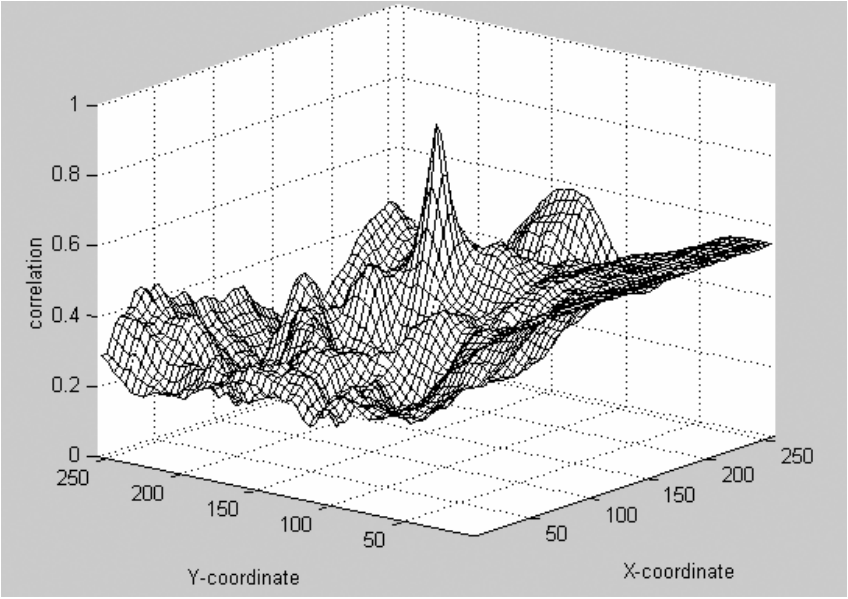

Figure 2.2: Correlation peak corresponding to the correct mapping between the reference image and the templates

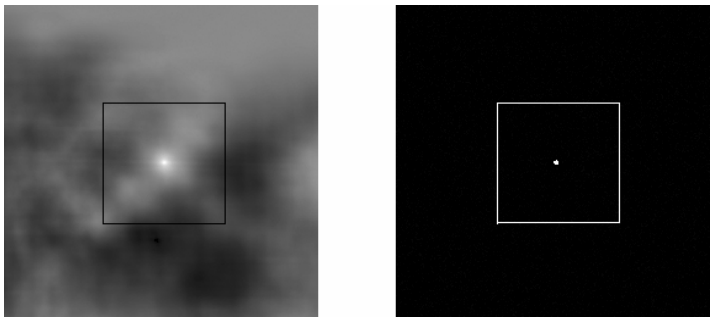

Figure 2.3: Cross correlation between the reference image and the undistorted templates

Figure 2.4 shows two distorted versions of the template image, where the distortion is achieved by stretching the image along the *x* axis (Figure 2.4, left), and along the *y* axis

(Figure 2.4, right), with the ratio 2:1. As shown in Figure 2.5, the cross correlation fails to align the reference and template images, when the latter has undergone the stretching distortion. The maximum value of the correlation function in this case does not correspond to the parameter values defining the correct alignment of the images. The correct location of the template image in the reference image is shown with a colored box.

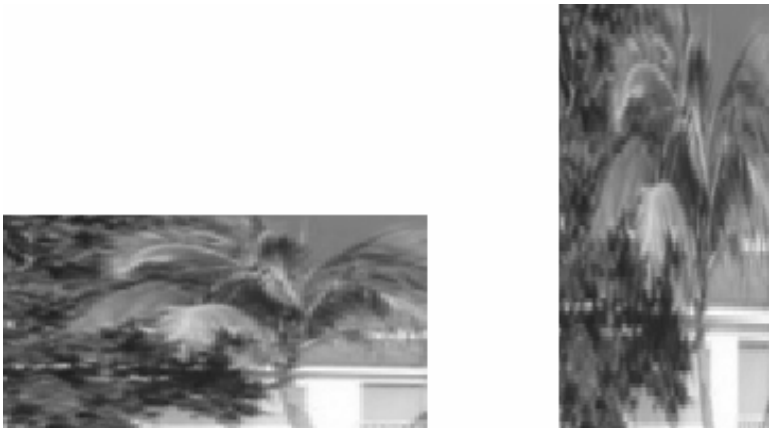

Figure 2.4: Template image stretched along the *x* (left) and *y* (right) axes

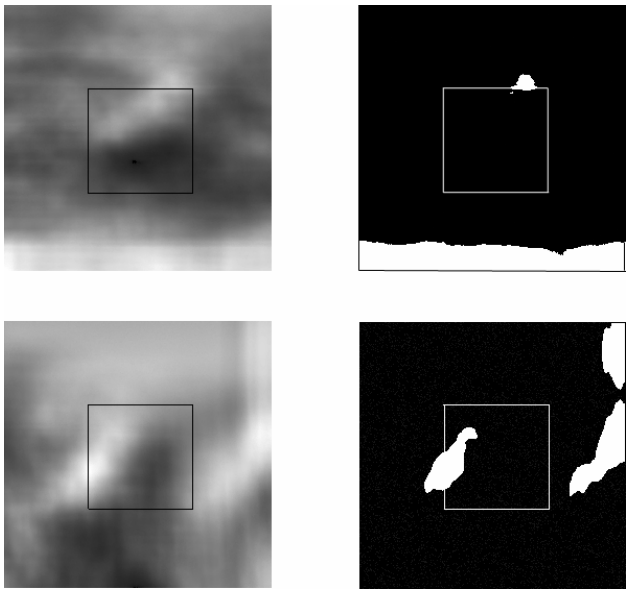

Figure 2.5: Cross correlation with *x*- (top row), and *y*-stretched (bottom row) templates

Fourier-based correlation of the undistorted and distorted template images is performed using Formula (2.3) and Fast Fourier Transform implemented in the MATLAB package [96]. The result of the Fourier-based correlation for the undistorted template image is presented in Figure 2.6, in the form of a 3-D plot of the normalized correlation function. Figure 2.7 shows the result of the Fourier-based correlation for the *x*-stretched template image. Figure 2.8 shows the result of the Fourier-based correlation for the *y*-stretched template image. As one can see from the 3-D plots of the normalized correlation function in Figures 2.7 and 2.8, Fourier-based correlation fails to correctly align the distorted images.

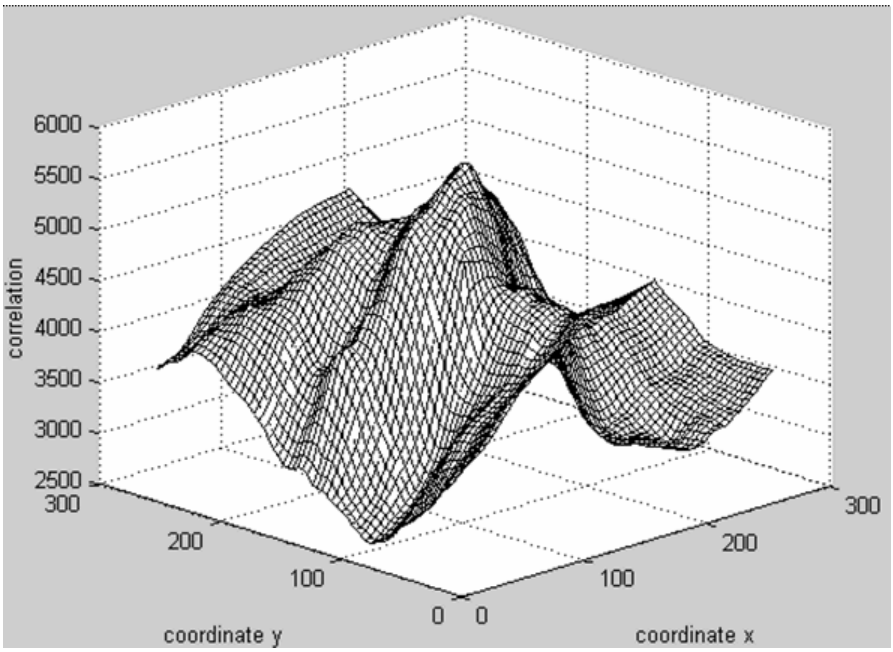

Figure 2.6: Fourier-based correlation with the undistorted template

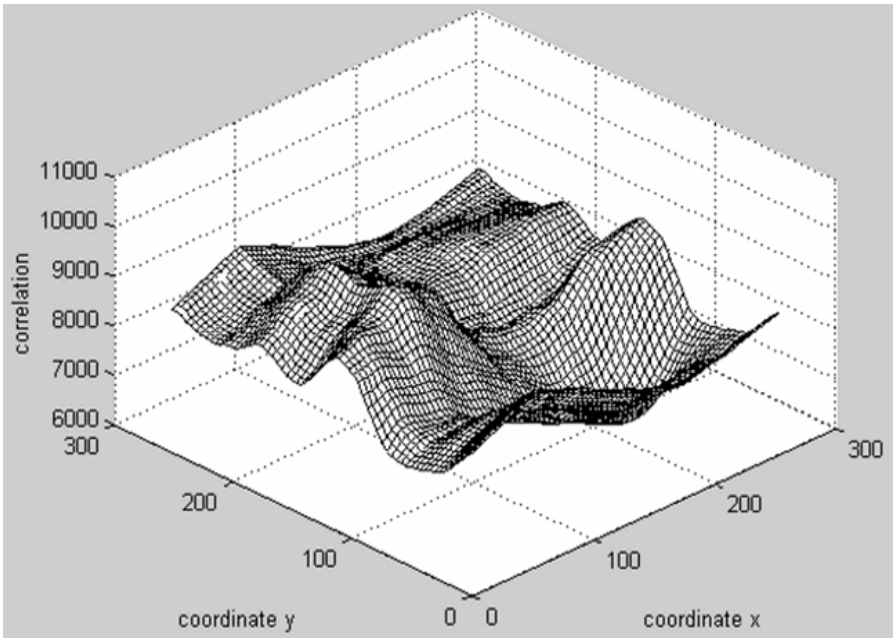

Figure 2.7: Fourier-based correlation with the *x*-stretched template

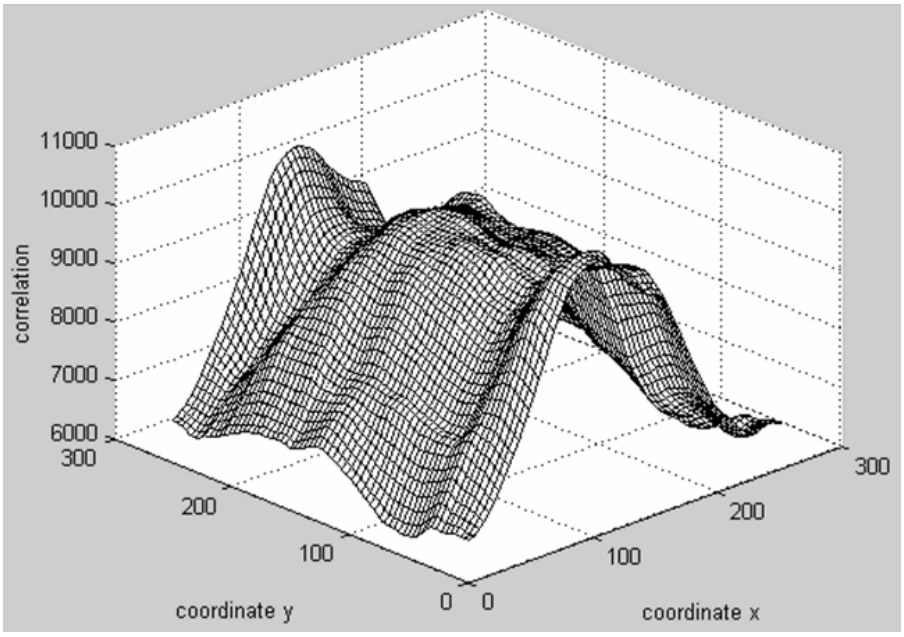

Figure 2.8: Fourier-based correlation with the *y*-stretched template

Image registration with mutual information considers any pixel location (x,y) within the image as a random variable, and the corresponding pixel value $P(x,y)$ as the outcome of the random variable (x,y). From the viewpoint of Information Theory, the pixel value P carries a certain amount of information defined as the logarithm of the probability $p(P)$ of the value P [62]. The expectation of the information carried by P is called entropy $H(P)$; it measures the level of uncertainty, or complexity of the random variable P, and is defined in [119] as

$$H(P) = -\int p(P)\ln(p(P))dP\,. \tag{2.4}$$

The joint entropy of two random variables P and Q is the property of their joint distribution $p(P,Q)$, and is defined in [119] as

$$H(P,Q) = \iint p(P,Q)\ln(p(P,Q))dPdQ\ . \tag{2.5}$$

Mutual information $I(P,Q)$ between two random variables is defined [119] via their entropies as

$$I(P,Q) = H(P) + H(Q) - H(P,Q)\ . \tag{2.6}$$

When a variable P is associated with the pixel distribution in the reference image Img_R, and a variable Q is associated with the pixel distribution of the template image Img_T, the first term in Formula (2.6) takes a fixed value, while the other two terms vary, in accordance with an applied geometric transformation T of the images. When both images are properly aligned, their mutual information takes its maximum value, i.e., the template image most effectively explains the reference image. The essence of image registration is

the maximization of the mutual information $I(Img_R, Img_T)$, while varying the transformation parameters; i.e., the registration problem is formulated as

$$T^* = \arg\max_T I(Img_R, Img_T), \quad (2.7)$$

where T^* is the correct transformation providing the proper alignment of the images Img_R and Img_T [173]. In fact, Formula (2.7) is the statement of an optimization problem. Two most common approaches to the maximization of mutual information are the exhaustive search throughout the parameter space, and gradient descent methods that use the derivatives of the mutual information.

The advantage of mutual information over the correlation techniques lies in its ability to register multi-modal images having different histograms, where direct comparison of the gray values is not possible. Mutual information is commonly used in medical imaging to register CT (computed tomography), MR (magnetic resonance), and PET (position-emission tomography) images. As with the correlation techniques, the use of mutual information is limited to the cases where:

- images are subject to similarity transformation, which includes translation, rotation, and isotropic scaling, and excludes a noticeable distortion of the images;
- search space of all feasible geometric transformations is relatively small, which allows for conducting an exhaustive search, or easily locating the maximum value with gradient descent methods.

Practical experience shows that there are certain difficulties in implementing and using mutual information:

- local optimization methods can perform poorly, and are prone to stalling at local optimum solutions;
- the success of the method significantly depends on the sampling strategy used in a particular application.

In order to test the behavior of the registration method based on mutual information, registration is performed for the original (undistorted) and distorted template images, with the help of the ITK Segmentation and Registration Toolkit [69]. The registration parameters are set to the following values:

- number of multi-resolution levels $N_L = 1$;
- number of iterations $N_I = 2500$;
- learning rate $R_L = 0.0001$.

Figure 2.9: Registration of the original template with mutual information: the output target (left), the output source (center), and the output registered image (right)

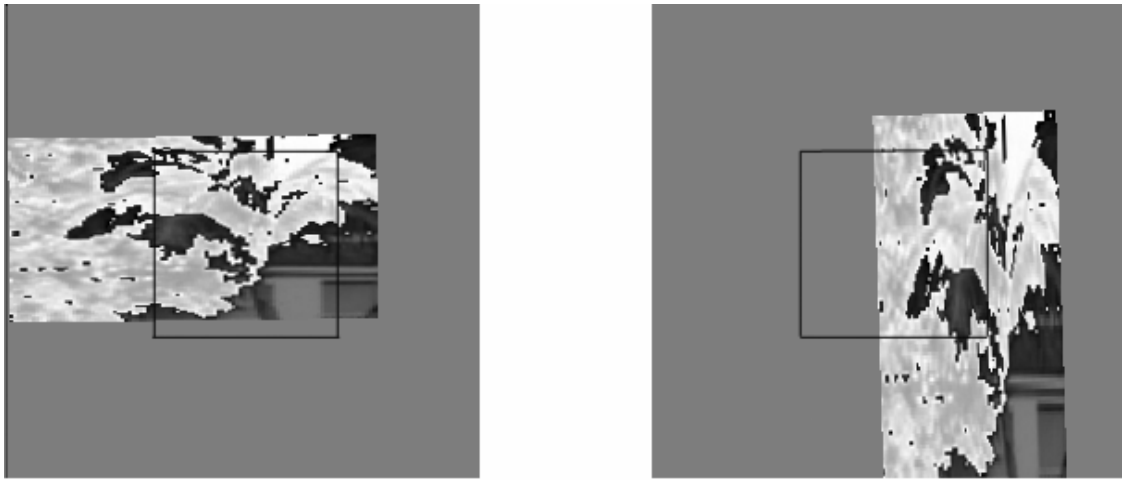

Figure 2.10: Registration of the distorted template with mutual information: the *x*- (left) and the *y*-stretched (right) registered images

Figure 2.9 shows the results of the registration of the original, undistorted template image, with the obtained translation parameters for the upper left corner of the image *DX* = 61.6 and *DY* = 102.7. The correct location of the template image is shown with a black colored box. For comparison, the correct values for the translations *DX* and *DY* are *DX* = *DY* = 80.0. Figure 2.10 shows the results of the registration of the distorted template images. The correct location of the template image is shown with a black colored box. The computed translation parameters for the upper left corner of the *x*-stretched image have the values *DX* = 1.2 and *DY* = 72.0. The computed translation parameters for the upper left corner of the *y*-stretched image have the values *DX* = 117.0 and *DY* = 64.4. As can be seen from Figures 2.9 and 2.10, the registration results are not satisfactory, particularly in the case of the distorted images.

In the cross entropy approach, the histograms of image pixel values are analyzed and compared, using entropy as a similarity measure. The difference $d(Img_R, Img_T)$ between

the pixel values of the reference and template images is computed for the particular (e.g., affine) transformation T. The gray-value histogram of the difference image $d(Img_R, Img_T)$ is built, and the entropy $H(T)$ of the histogram is computed as

$$H(T) = -\sum_{P=P_{\min}}^{P=P_{\max}} p(P) \log p(P) \ , \tag{2.8}$$

where $P_{\min}$ and $P_{\max}$ are the lower and upper pixel values, respectively, and $p(P)$ is the fraction of pixels with the gray value P in the histogram H [14].

Misregistration of images results in a broad histogram, while properly aligned images produce a histogram with a few peaks associated with the features of the images. Consequently, the entropy has its minimum value when images are properly aligned, i.e., the registration task can be stated as a minimization problem in the space of all feasible image transformations. The search for the optimum transformation T^* is usually performed using a hill-climbing/descending procedure.

Figure 2.11: Difference between correctly aligned reference and template images: the undistorted (left), the *x*-stretched (center), and the *y*-stretched (right) templates

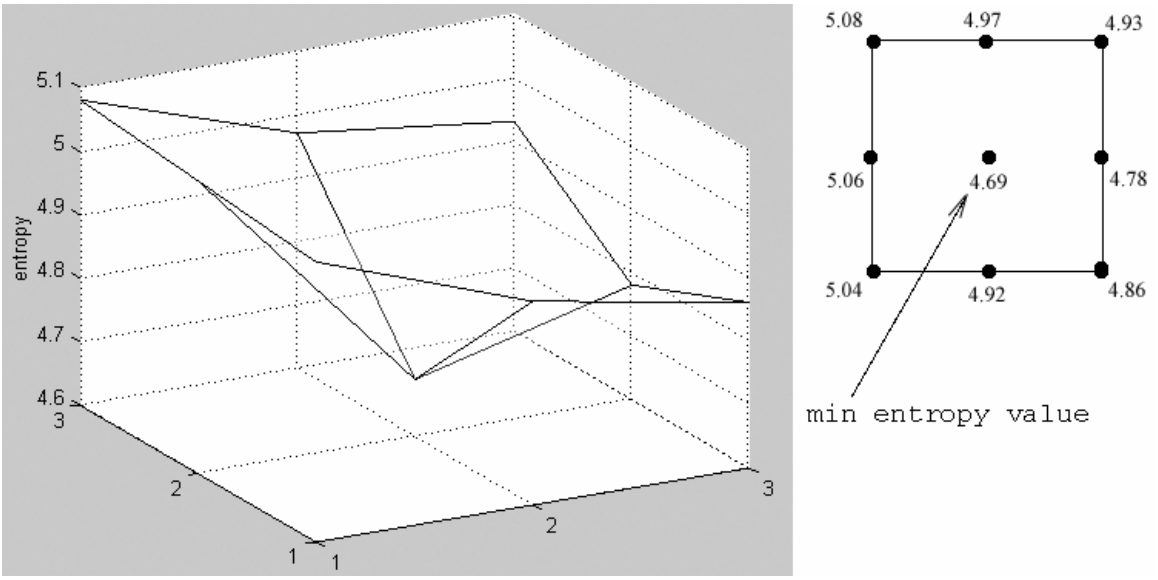

Figure 2.12: Entropy values for the undistorted template: the 3-D plot (left) and the corresponding 2-D matrix (right), for different values of the translation

Figure 2.11 shows the difference $d(Img_R, Img_T)$ between the reference and original template images; and between the reference and distorted template images, when the respective template images are correctly aligned. The entropy value for the original, undistorted image is computed for the correct alignment, and for the translations by 10 pixels along the *x* and *y* coordinates. In Figure 2.12 (right), the central point corresponds to the correct alignment, while the other eight points correspond to the translations of the image in the corresponding directions. As one can see, the correct alignment of the test image at the central point results in the minimum value of the entropy $H(T^*) = 4.69$. The entropy grows, as the image continues to shift from its correct location; the value of the entropy varies in the range of $H(T) = 4.78$ through $H(T) = 5.08$, for the incorrect alignment.

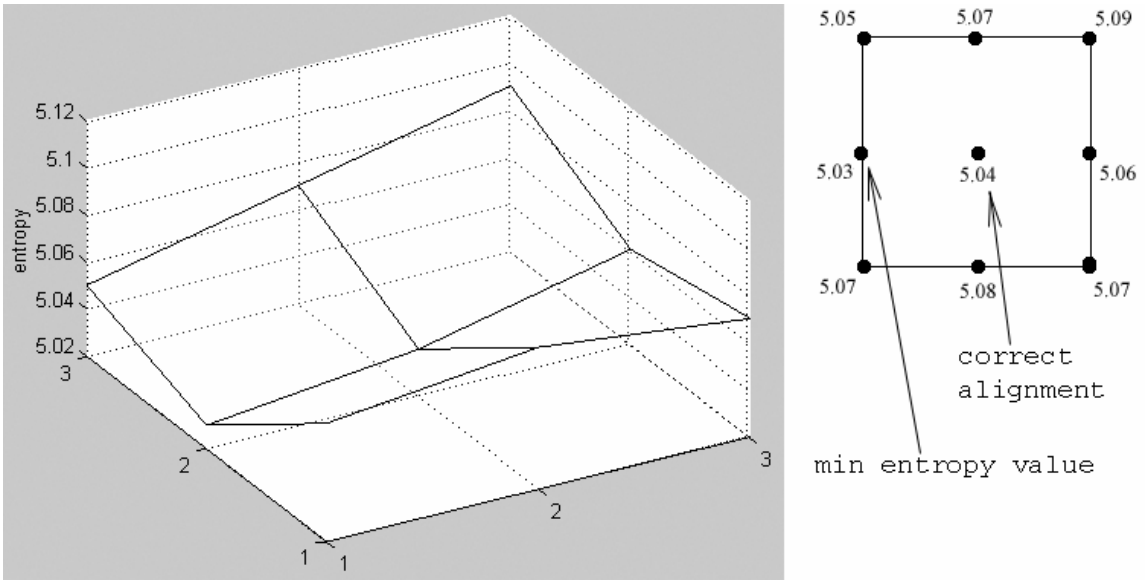

Figure 2.13: Entropy values for the *x*-stretched image: the 3-D plot (left) and the corresponding 2-D matrix (right), for different values of the translation

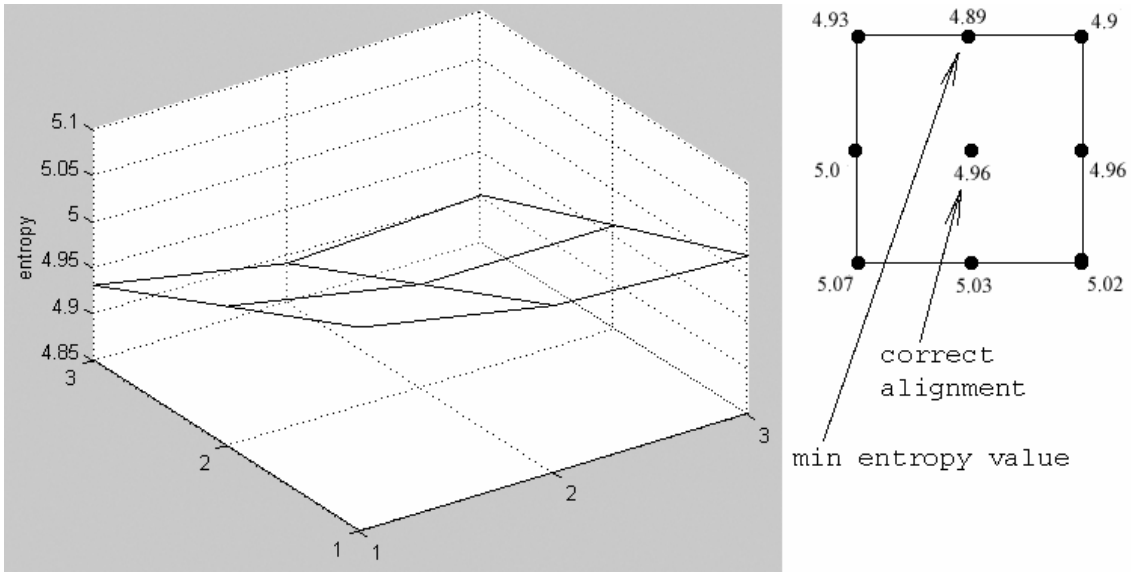

Figure 2.14: Entropy values for the *y*-stretched image: the 3-D plot (left) and the corresponding 2-D matrix (right), for different values of the translation

Figure 2.13 shows the entropy value for the *x*-stretched template image: the minimum value $H(T) = 5.03$ corresponds to the translation parameters $DX = 120$ and $DY = 130$; i.e., the minimum is shifted to the left from its correct value $DX = 130$. In Figure 2.14, the minimum entropy value $H(T) = 4.89$ for the *y*-stretched template image corresponds to the translation parameters $DX = 130$ and $DY = 120$; i.e., the minimum is shifted upward from its correct value $DY = 130$.

2.1.2. Two- and Three-Dimensional Non-Rigid Registration

In non-rigid, or elastic, registration, deformation of the images is allowed, in addition to the rigid body transform. The type of the transformation, the similarity measure between the images, and the optimization method used to find the correct transformation parameters define the specifics of different methods of non-rigid registration. Most research on non-rigid registration has been done in medical imaging, particularly related to brain research.

Large group of methods designed for non-rigid registration requires establishing reference, or control points by human operator, which makes these methods unsuitable for autonomous systems. Much more interesting are methods that attempt to automatically establish the correspondence between image pixels, and to find the correct image mapping. One of the commonly used methods of non-rigid registration is the Iterative Closest Point (ICP) algorithm [10, 18, 136]. The algorithm is fairly versatile, and can be applied to the tasks of matching sets of scattered points, as well for matching free-form curves and surfaces. ICP is used for such problems as image registration, 3-D

surface alignment, and motion estimation. Many variants of ICP have been proposed in the literature, different from each other in the way they affect the various phases of the algorithm - from the selection of the matching points, to the minimization strategy [136].

Below is the description of the algorithm for matching the data set P with the model surface S, according to [10]. The matching problem is re-formulated in terms of deformation $F(\boldsymbol{q})$ defined by the vector of registration parameters $\boldsymbol{q}$, and applied to the data set, in order to obtain the surface S, i.e.

$$S_i = F(P_j) \,, \tag{2.9}$$

where S_i is the point of the model surface obtained after the deformation $F(\boldsymbol{q})$ has been applied to the point P_j of the data set. The best deformation function F defined by the vector of registration parameters $\boldsymbol{q}$, has to be found that minimizes a chosen cost function, usually the averaged distance between the corresponding points of the surfaces. The algorithm alternates the phase of finding the correspondence between the points in the data set P and the points on the surface S; and the phase of finding the deformation F, i.e., the registration parameters $\boldsymbol{q}$. The convergence theorem states that ICP always converges to a local minimum, with respect to the mean-square distance chosen as a cost function [136]. Local optimization techniques are used to minimize the cost function at every iteration, in order to find the new approximation to the deformation F. In practice, it means that the ICP algorithm can be applied only when the deformation F or, alternatively, the vector of the registration parameters $\boldsymbol{q}$ is small, i.e., the surfaces that have to be mutually registered, are close enough to each other.

2.1.3. Conclusions on Commonly Used Methods of Image Registration

The vast majority of the existing methods of image registration can deal with the rigid body or similarity transformations between images subject to registration, in a fairly robust way. Fewer methods have been developed to deal with the geometrically distorted images. The methods in this category either assume that distortion is small enough, or impose certain restrictions on the type of images, or on the type of transformation between the images. There are no robust methods of image registration that would be general enough to handle global image mapping including both, the similarity transformation and the arbitrarily large image distortion.

2.2. Object and Target Recognition

Object recognition, as well as closely related target recognition, is important part of many visual applications, particularly robotic and computer vision; security; and defense. By and large, object recognition (OR) systems have to localize, identify, and classify different objects in the scene, as well as to provide the necessary information about the objects, e.g., the transformation parameters like rotation angle, scaling factor, and others. For example, in the typical scenario of the robotic vision, robot is shown the template images (or models) of the objects of interest. Robot has to find then the objects in the real scene, where it moves by comparing the objects present in the scene with the learned templates.

Different methods have been developed to solve object recognition problem. Among the commonly used methods are those based on image correlation, mutual information,

entropy measure, and Artificial Neural Networks (ANNs). The first three categories of the methods are discussed in section 2.1 of this Chapter. The Artificial Neural Network approach is discussed in this section.

In its most general form, the ANN-approach includes three phases:

- choosing the network architecture and parameters, and building the network;
- training the network on one or more training sets;
- recognizing the objects using the designed network.

Numerous methods utilizing ANNs differ in architectural principles, mechanisms, and techniques used for the first two phases. As an example, one of the interesting and more advanced algorithms developed in [28, 29, 43, 157, 158, 163, 164, 165] is discussed in this section. The algorithm is based on large experimental work on the primate vision system, and is implemented in the form of the software package SpikeNet simulating the activity of very large neural network of asynchronously firing neurons. The basic concept of the algorithm is the rank order coding that selectively favors the order in which the artificial neurons fire. For every image of a scene, a multi-layer map of artificial neurons is generated, of a size of the image bitmap; the map simulates the processes that occur in the human or monkey visual system. The response of the different neurons varies with the strength of the input stimulus. The neurons that receive stronger stimulus reach the threshold more quickly and fire first; they are given higher weights in the network.

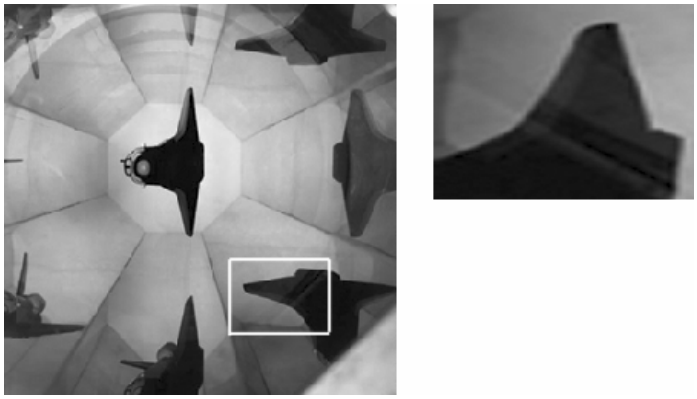

Figure 2.15: Test images: the original scene (left) and the sought distorted object (right)

In order to evaluate the ability of the algorithm to recognize significantly distorted objects, the SpikeNet-based demo version of the commercially available package implementing the algorithm, SNVision Model Builder from Spikenet Technology, is tested [151]. The set of the test images is shown in Figure 2.15 [70]. The set includes the 256×256-pixel scene containing several objects (left), and the 170×128-pixel image of the sought object (right) obtained by cropping a section from the scene. The image of the object undergoes an affine transformation defined by the 5-dimensional vector $\boldsymbol{V} = \{DX, DY, \theta, SX, SY\}$. The translations DX and DY, and the rotation angle θ define the position of the object in the scene (the rigid body transformation), while different scaling factors SX and SY define the local distortion of the object (the local transformation). The object is significantly distorted, i.e., it is stretched along the x axis, with the ratio $SX / SY = 2$. The correct values of the components of the parameter vector are $\boldsymbol{V}^* = \{150, 212, 1.57, 4.14, 2.07\}$.

The image of an aircraft wing in Figure 2.15 (right) is an interesting and computationally hard recognition problem. The six projections of the same object are considered here as

six different objects in the 2-D scene. The objects have similar shapes, but only one object is the actual (i.e., the correct) solution. The purpose of the experiment is to test the ability of the SNVision Model Builder to distinguish between the similar objects. The program has to identify the correct object via investigating possible combinations of the global and local transformations, for all objects in the scene, and via detecting the combination and the object that yields the best match.

SNVision Model Builder starts with training the network with the template image (i.e., the model) of the object, for a range of rotation angles and scaling factors. As an example of the object model and the result of the object recognition, a demo image provided with the package is tested. The model of the bolt nut is trained for the range of the rotation angle θ = (-12°) through (+12°), with the step size 6°, and for the range of the scaling factor $SX = SY$ = 90% through 110%, with the step size 10%. With the parameters of the recognition network set to the medium quality, the algorithm is able to recognize all objects in the scene.

For the recognition of the object shown in Figure 2.15 (right), firstly, the recognition quality is tested on the template of the object. During training, the model of the object is built for the rotation angle in the range θ = (-180°) through (+180°), with the step size 10°; and in the range of the scaling factor $SX = SY$ = 20% through 100%, with the step size 10%. The recognition quality is set to the medium level. The algorithm is able to correctly recognize the object – see Figure 2.16. The correctly recognized object is indicated with a colored box on the left image.

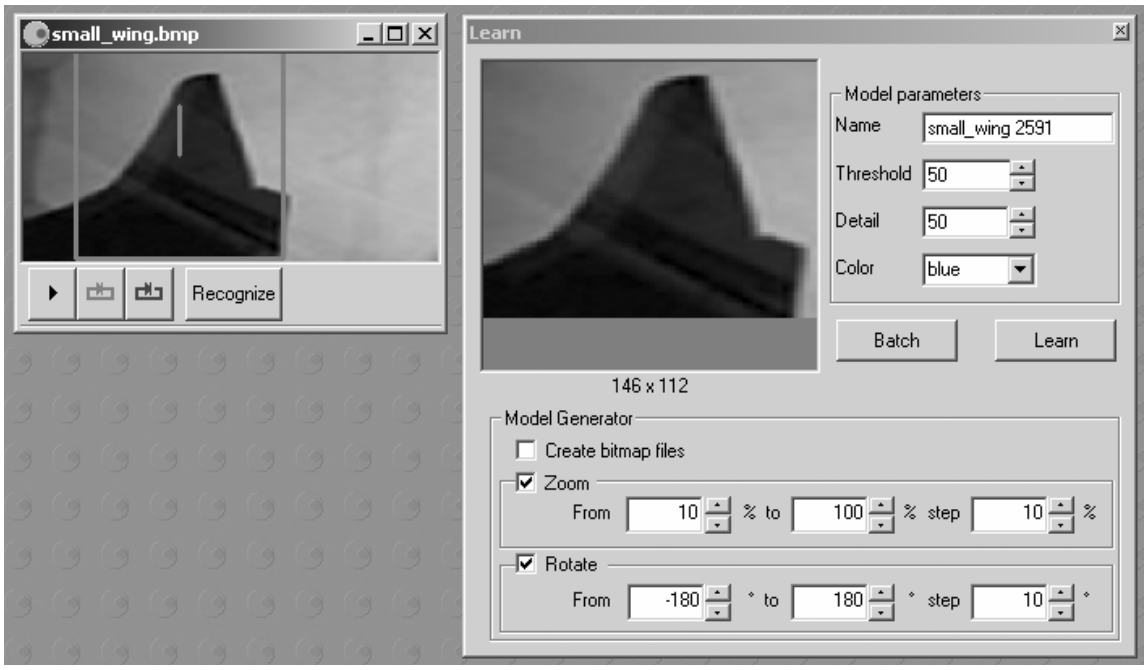

Figure 2.16: Result of the recognition (left) of the template model (right)

For the actual recognition of the wing object in the entire scene shown in Figure 2.15 (left), the same model that has been built in the previous experiments is used, but the quality of the recognition is varied, from medium to high. The typical result for the high quality recognition is shown in Figure 2.17. The falsely recognized objects are indicated with colored boxes on the left image. The algorithm is not able to find the correct object in the scene; it either points at the wrong location, or at the wrong object. The poor performance of the ANNs can be attributed to the fact that they conduct local search; consequently, they can fall into one of the local minima, in the case when the objective function is multi-modal, i.e., contains multiple local minima.

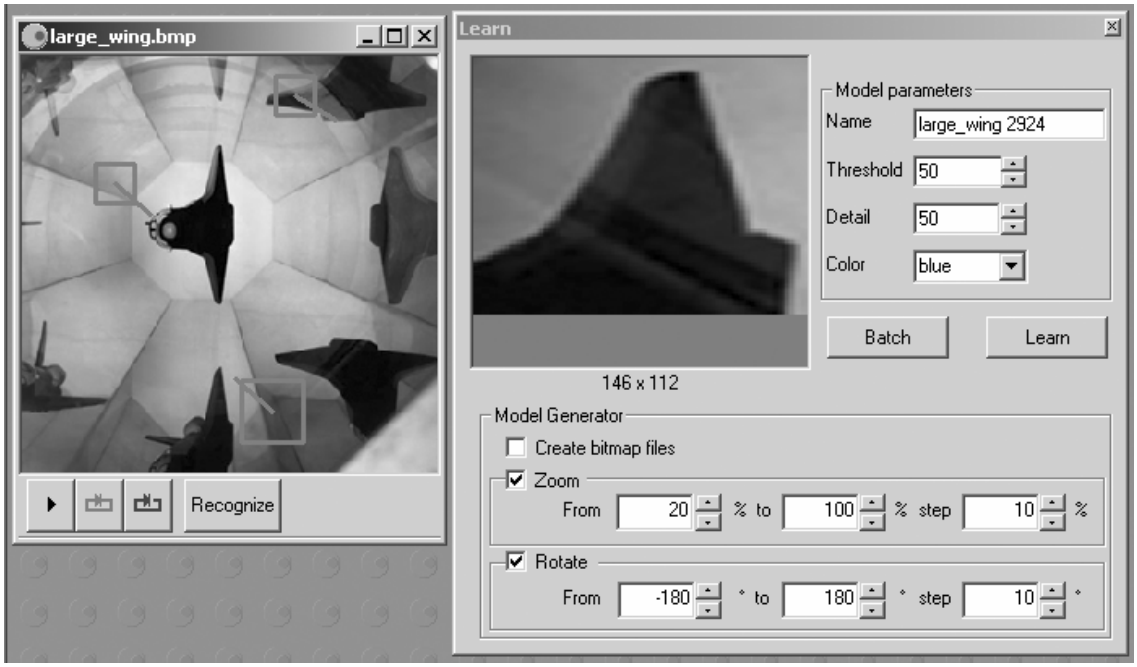

Figure 2.17: Typical result of the recognition (left) of the template object (right)

Although Artificial Neural Networks are widely used in object and target recognition, they have the following problems:

- networks require an exhaustive training for all feasible ranges of the transformation parameters.
- networks often fall into local minima during the search for an object in a scene, resulting in false recognition.
- networks require re-training when the type of the problem changes, e.g., when the object template is modified.

2.3. Content-Based Image Retrieval

Image search and retrieval from a large image database is an active research area that has a multitude of sub-areas and approaches [79]. For example, the focus can be on finding a

particular object that can be present in images of the original scene stored in a database. The original scene might be cluttered, and might contain images of different objects. Moreover, the object in the query image can be distorted, e.g., by affine or perspective transformation, which makes methods based on correlation or some measure of feature similarities between the images less efficient.

The main underlying principles of the content-based image retrieval system QBIC developed by IBM are discussed in [33, 117]. The following image features are used to store, search, and retrieve images from the database: color, texture, and shape. Finding objects having a similar shape to the query image is one of the most difficult problems, in authors' opinion [117]. The system includes a number of heuristic shape features, including area, circularity, eccentricity, axis orientation, and algebraic moment invariants. All shapes have to be planar and non-occluded; they are represented as binary images. In addition, an edge map based on Canny edge extraction algorithm, can be used for a search, when the image is represented in the form of a sketch. The attributes of the database images are pre-computed, and stored as multi-indexed records. The operation of search and retrieval is implemented as the search through the indexed records, followed by the similarity-based retrieval of the images, where similarity is defined in terms of Manhattan, Euclidian, or weighted Euclidian distance.

Although content-based image retrieval systems belong to the area of extensive research, the difficulty of the shape recognition problem in general, and similarly-shaped image retrieval, in particular, lead to approaches that attempt to avoid the direct global search on

the database images, and engage pre-processing image analysis, followed by indexing of the image records.

The problem of content-based image retrieval can be put into a more general, global optimization framework, which is free of constraints imposed by the type of transformation presumably existing between the images. Given two images, the image of the scene Img_0, and the image of the object Img_1 (the query image), the task of finding the potential matches between Img_1 and objects in Img_0 can be formulated as the following global optimization problem. If image Img_1 undergoes transformation $A(\boldsymbol{V})$ defined by the parameter vector $\boldsymbol{V}$, then the optimum vector $\boldsymbol{V}^*$ has to be found that minimizes the difference F between the images Img_1 and Img_0. The algorithm that implements the search and retrieval task, has to search through the database, retrieve the stored images Img_0, solve imaging optimization problem for the images Img_1 and Img_0, and offer for user evaluation those images Img_0 that provide the feasible transformation $A(\boldsymbol{V})$, and the minimum value of F.

2.4. Summary

Many commonly used methods of automatic image registration; object and target recognition; and content-base image retrieval satisfactorily work on images that undergo the rigid body, or similarity transformations. Methods like correlation techniques, mutual information, cross entropy, iterative closest point algorithm, and Artificial Neural Networks frequently show poor performance, when images are significantly distorted e.g., by affine or perspective transformation. More general approaches are required that

are free of limitations imposed by the current methods on the type of imagery, or the type of transformation. One of the potentially advantageous approaches involves using stochastic methods of global search and optimization, particularly methods comprising the family of Evolutionary algorithms.

Chapter 3: Overview of Evolutionary Algorithms in Global Optimization and Search

3.1. Foundations of Evolutionary Algorithms

Evolutionary algorithms (EAs) belong to the category of global search and optimization methods, and are inspired by the principles of natural evolution and genetics. Just like other modern heuristics like Simulated annealing and Tabu search, Evolutionary algorithms draw from observations of physical processes that occur in nature, and attempt to mimic them in the artificial environment. This section gives a brief overview of the theoretical foundations of Evolutionary algorithms.

3.1.1. Introduction

Evolutionary algorithms have interdisciplinary character; they bring together the areas of biological evolution, global optimization, and Artificial Intelligence (AI) [4, 12, 57]. Reportedly, EAs work well on nonlinear, non-convex, non-separable, discontinuous, non-differentiable, multimodal, noisy, and other computationally hard real-world problems, where traditional optimization methods frequently fail or perform poorly. According to [61, 66], the algorithms have advantage over their traditional counterparts, under the following conditions:

1. The search space has high cardinality (dimensionality). The optimization problem is highly nonlinear, which makes it impossible to build a precise analytical model.
2. The set of potential candidate solutions is extensive, which makes the enumerative search (i.e., brute force approach) intractable.

3. Parameters that have to be optimized can be discrete or continuous, and the knowledge of their derivatives is not available or required. The data can be generated experimentally, numerically, or analytically.
4. Knowledge of the common properties of the higher-than-average performing subsets of candidate solutions is incomplete. This makes it difficult to infer the robust way of determining what specific properties lead to higher performance.
5. Performance of a candidate solution is highly dimensional and nonlinear, and exhibits local sub-optimal areas, where the search can be trapped for undetermined period of time.
6. Performance of candidate solutions is a random variable or process whose observed behavior is subject to sampling errors. Therefore, the search algorithm needs to maintain a proper balance between exploitation of the already known properties, and acquisition of new information.

When compared to other methods that deploy random search, Evolutionary algorithms have winning points [66], due to the following intrinsic qualities:

- Progressive exploitation of the best-observed candidate solutions.
- Increasing confidence in the estimate of the expected performance of the best-observed candidate solutions, as the algorithm progresses.
- Implicit parallelism in testing a large number of combinations of potential optimal subspaces, including both, newly generated and already tested high-performance samples. The term “implicit parallelism” refers to the simultaneous processing of all possible combinations of partial effective gene structures called schemata.

3.1.2. Biological Background of Evolutionary Algorithms

Evolutionary algorithms borrowed much of their terminology from biology, since the main ideas of the algorithms were originally inspired by the results and achievements in biological sciences [61, 66, 111]. Every biological organism consists of elementary building blocks, or cells. Every cell carries a set of *chromosomes* – strings of DNA that store the most important features of the organism. A chromosome is divided into *genes* – smaller DNA blocks, each responsible for a particular feature (*trait*), and located in a particular place on a chromosome, called *locus*. Different settings for a trait are called *alleles*. Genes can interact with each other, so that gene responsible for one trait can affect genes responsible for other traits.

The complete set of chromosomes is called *genome*. Organisms that have identical genomes belong to the same *genotype*. The genotype to a large extent determines the organism's *phenotype* – its observed particular physical and mental characteristics. Adaptation of the organism is primarily a search for *co-adapted* sets of alleles, i.e., combinations of alleles in different genes that significantly improve the performance of the organism in a particular environment. The success with which the individual adapts to the environment is measured by the organism's *fitness*.

In order for evolution to take place, the transmission of genetic material has to be simultaneously stable and variable. Stability ensures that good parental traits would be likely transmitted to offspring by means of reproduction. The latter is carried out in the heterosexual population via *crossover* (*recombination*). In order to introduce dynamics

into population, some external factors have to be in place that will provide a chance of transmitting new or different traits. One of such factors is *mutation*, a random change in genes that can lead to significantly different phenotype of the offspring compared to its parents. Another factor is *gene flow* resulting from the embedding of a new organism into the breeding population. The third factor is *genetic drift*, where certain alleles can be lost during random recombination in small populations. The last and important factor is *natural selection*. During selection, the best (i.e., the fittest) organisms are chosen to be parents, i.e., certain best characteristics are selected into the breeding pool.

Following the biological terminology, Evolutionary algorithms typically use the term *chromosome* to denote a single candidate solution to a problem. Smaller blocks that constitute the solution correspond to genes. For example, in multidimensional optimization, a chromosome is a vector of parameters that constitute a potential solution, while a gene is a single component of the vector. An allele is a particular value of the component or its part. Fitness typically corresponds to the objective function of the optimization problem, or some form of its surrogate. In order to mimic biological reproduction between two single-chromosome parents, Evolutionary algorithms select parental chromosomes from the current population, according to their fitness. The fitter the chromosome, the higher is its likelihood of being selected as a parent. Crossover chooses a locus at random, and exchanges the bit substrings before and after the locus, to form two offspring. Mutation is implemented by occasionally flipping some of the bits in the offspring with a certain (usually small) probability.

Evolutionary algorithms attempt to solve optimization problem by searching for a chromosome (i.e., parameter vector) that has the minimum (in minimization problem) or the maximum (in maximization problem) value of its fitness function. The algorithms apply various genetic operators (most often, selection, crossover, and mutation) to the current population of chromosomes, in order to produce the new population (i.e., the next generation). The algorithms repeat the process until a satisfactory optimal solution is found.

3.1.3. Classical Representatives of Evolutionary Algorithms

The term "Evolutionary algorithm(s)" covers a spectrum of independently developed mainstream methods: Genetic algorithms (GAs), Evolution strategies (ES), Evolutionary programming (EP), and Genetic programming (GP) [37]. Although originated independently, these methods have fundamental commonalities, like using reproduction, random variation, competition, and selection of competing individuals. Table 3.1 summarizes the main characteristics of the four classical methods, as they were originally formulated. The rest of the section provides a brief discussion of each method.

The coherent concept, and the general theory of Genetic algorithms was developed by J. H. Holland, as the result of studying natural adaptation of living organisms, and ways of incorporating a similar adaptation into computer systems [66]. Holland looks at adaptation in terms of the environment E in which the system has to survive, the adaptive plan τ that determines successive structural modifications in response to E, and some measure μ of the performance of the adaptive system. The objective of the plan τ is to

produce the structures showing good performance μ, in response to the current environment E. The performance is measured in terms of the objective of the search. The adaptive plan has to be robust with respect to retaining the higher-fit structures found to date, as well as increasing their proportion in the population. The plan repeatedly acts on the population of attainable structures, as a two-phase procedure. During the first phase, additional copies of the higher-fit individuals are added to the population, in place of the deleted lower-fit individuals. The number of added copies of the higher-fit individuals is proportional to their performance (fitness). During the second phase, genetic operators W are applied to individuals, interchanging and modifying them. The result is a new, modified population. The process continues producing a new generation at every step.

The concept of Evolution strategies (*Evolutionstrategie*) was introduced by I. Rechenberg and H.-P. Schwefel [129, 130, 142]. The idea was to employ evolution as a tool for solving real technical problem, in particular, evolving the airfoil geometry. The problem is stated as the minimization of the functional $f(x)$, where x is the n-dimensional vector. A potential candidate solution is represented as a fixed-length, real-valued organism (i.e., chromosome); it is initialized as a parent from the feasible solution space, at random. An offspring x' is created from the parent x using Gaussian mutation, i.e., by adding a Gaussian random variable with zero mean and a pre-defined standard deviation to each component of x. The best individual is selected from the pair parent-offspring using ranking, with respect to the value of $f(x)$. The selected organism becomes the parent for the next generation. The evolutionary process continues until either a satisfactory optimum solution is found, or the available computation time is exhausted.

Algorithm attributes	Algorithm			
	GA	**ES**	**EP**	**GP**
Population	Chromosomes	Pair parent-offspring	Finite-state machines	Executable programs
Representation of solution	Binary encoding	Real-valued encoding	State-transition diagram	Tree structure
Encoding length	Fixed	Fixed	Variable	Variable
Evolutionary operators on solutions	Selection, crossover, mutation	Selection, mutation	Selection, mutation	Selection, crossover, mutation
Selection method	Fitness-proportional	Ranking	Ranking	Ranking
Object's fitness evaluation	Value of the objective function	Value of the objective function	Result of symbol prediction	Result of program execution
Control parameters	Fixed and pre-set	Self-adaptive	Fixed and pre-set	Fixed and pre-set

Table 3.1: Main characteristics of the classical representatives of Evolutionary algorithms

Later, multiple parents and multiple offspring were introduced into the ES framework, and two main approaches known as $(\mu+\lambda)$-ES and (μ,λ)-ES were developed [143]. In the first, μ parents are used to produce λ offspring. All $(\mu+\lambda)$ solutions compete for survival,

but only μ best are chosen as parents for the new generation. In the (μ,λ)-approach, only λ offspring are competing, and μ of them completely replace the current population. The current status of Evolutionary strategies extends comma- and plus-schemes to a more general (μ,k,λ,ρ)-scheme [144]. The scheme introduces the maximum life span of $k \geq 1$ generations, and an arbitrary number of ancestors $\rho \geq 1$ for each descendant. Comma- and plus-schemes are the special cases of the extended scheme, when $k = 1$ and $k = \infty$, respectively.

Evolutionary programming was developed by L. J. Fogel, A. J. Owens, and M. J. Walsh [38, 40, 41], as an approach to artificial intelligence, based on adaptive behavior to meet goals in a range of environment. A population of candidate solutions, in a form of small variable-length finite-state machines represented by their state-transition diagrams was exposed to the environment. For the sake of generality, the environment was presented in a form of a sequence of symbols taken from a finite alphabet. For every machine, an input symbol was presented, and the output symbol was compared with the next input symbol. The result of the prediction was measured in terms of a payoff function (e.g., the squared error). After all the predictions had been made, the fitness of the machine was evaluated as the average payoff per symbol. Finite-state machines evolved by randomly mutating their state-transition diagrams, and selecting the fittest (i.e., having the greatest payoff) machines to become parents for the next generation. The process continued until an entirely new symbol was met that did not belong to the environment. The best machine was selected to predict the symbol; the symbol was added to the environment, and the process re-iterated.

Genetic programming was introduced by J. R. Koza [88]. Genetic programming is a particular form of evolutionary computations, in which the evolving data structures are executable computer programs. A program is presented in a form of a variable-length tree structure. GP uses genetic operators of crossover and mutation, in order to evolve the candidate solutions. The fitness value of the program is evaluated by executing the code and evaluating the results. Therefore, GP is the evolution-driven search for a program that produces a desired output, in the space of all executable programs. One of the most distinctive properties of GP is the variable length of the representation of a candidate solution.

In the 1990s, there was a growing interaction between researchers working in different areas of evolutionary computations. As a result of this interaction, ideas were borrowed, exchanged, and modified across all areas. The boundaries between Evolution strategies, Evolutionary programming, Genetic algorithms, and Genetic programming have been fading. The current state-of-the-art suggests using the term *Evolutionary algorithm(s)*, to refer to all algorithms that use population-based random selection and variation [109]. This work will follow this suggestion, except for the cases when the original terms like *Genetic algorithms* are used due to historic reasons.

The prototype of Evolutionary algorithms has the following general characteristics [5, 26] – see Table 3.2:

1. At each iteration, EAs use the collective learning experience of the population or sub-populations, rather than that of a single solution. The individual candidate

represents a point in the search space, and can incorporate certain strategy (or control) parameters of the search. Working with a number of solutions, EAs can find multiple optimal solutions in a single step.

2. A well-defined fitness of the individual serves as a measure of its quality and chances to survive in a given environment. The selection process for the next generation favors the individuals having better fitness values.
3. Offspring are generated from the selected parents by means of a randomized process that includes some form of mutation and recombination (or crossover). Mutation mimics errors in the reproduction process of a single individual, while recombination exchanges information between two or more individuals.
4. No gradient information is used in the EAs operators, and solution representation is flexible. These properties make EAs flexible enough to apply to a wide variety of problems. EAs use stochastic principles, and do not assume any particular structure of the optimization problem.

The EAs formulation of a particular optimization problem includes an appropriate problem-dependent representation of a candidate solution, in terms of genes and chromosomes. Fitness function of a chromosome has to be defined according to the goal of the search (e.g., the objective function). The correct statement of the problem significantly contributes to the overall success of the solution obtained with Evolutionary algorithms. Once the problem has been stated, a simple Evolutionary algorithm with binary encoding, fitness-proportional selection, and single-point crossover

(corresponding to the classical Genetic algorithm) works as an iterative procedure [111] shown in Figure 1.1 of Chapter1.

Target objects	Population or a group of sub-populations of chromosomes
Digital representation of individual solution	Binary, real-valued, or any other general structure encoding
Encoding length of individual solution	Fixed or variable
Evolutionary operators on objects	Selection, crossover, mutation, and other problem-dependent operators
Object selection method	Fitness-proportional, ranking, and other methods favoring fitter individuals
Object's fitness evaluation	Objective of search or optimization, or any kind of its surrogate
Settings of control parameters	Pre-set, adaptive, or self-adaptive

Table 3.2: Main attributes of the EA prototype

The entire set of iterations (or generations) of the algorithm is often called a *run*. It is common practice to terminate the process after a certain number of generations, or after a satisfactory result has been produced. At the end of the process, one or more chromosomes are obtained that have desired optimum (minimum or maximum) values of fitness function. Simple Evolutionary algorithm serves as the reference model in the most theoretical work on evolutionary computations, as well as the basis for numerous EAs

variants and applications. Various flavors of EAs differ in representation of a candidate solution, in relative significance and types of evolutionary operators, in the number of parameters that control the search, etc.

3.1.4. Theoretical Foundations of Evolutionary Algorithms

A great deal of research has been done on theoretical justification of Evolutionary algorithms. Three main directions on which theoretical work has been focused include analysis of convergence and convergence rate of EAs, finding what problems are hard for EAs, and studying the computational complexity of the algorithms [179]. The overview given in this section serves only as a brief outline of the results that have had a significant impact on the development of EAs, with the emphasis on its most general version, Genetic algorithms. Theoretical concepts help better understand the fundamental properties of Evolutionary algorithms, and serve as the guiding principles for the algorithm implementation and variations.

The notions of "schemata" and "building blocks" are at the heart of the classical theoretical basis of Genetic algorithms [48, 66]. A schema is a set of bit strings described by a template, for example, the schema H = 111*****1 describes the 9-bit strings that have the first three bits and the last bit equal to 1, while the asterisk represents any binary value (a wild card, or the "don't care" symbol). A building block (BB) is a low-order, short-length schema. The idea of building blocks is that a good solution is made up of good combinations of bit values that provide the strings with better fitness.

Over the course of producing new generations, some of the schema instances can increase, while others can decrease in number. The dynamics of the process can be approximately estimated using the following Formula [48, 66], known as the Schema theorem:

$$E(m(H,t+1)) \geq m(H,t)\frac{u_m(H,t)}{f_m(t)}\left(1-p_c\frac{d(H)}{L-1}\right)(1-p_m)^{o(H)}, \quad (3.1)$$

where $E(m(H, t+1))$ is the lower bound for the expected number of the instances of the schema H at the time $(t+1)$, $u_m(H, t)$ is the average fitness of instances of H at the time t, $f(x)$ is the fitness of the string x, $f_m(t)$ is the average fitness of the population at the time t, $d(H)$ is the defining length of the schema H in the L-dimensional search space, p_m denotes the probability that any bit in a string x is mutated, and $o(H)$ is the order of H.

Another fundamental concept in the GAs theory, the Building block hypothesis, emphasizes the role of crossover as the major source of the GAs power [47, 48, 66]. The hypothesis states that GAs use crossover to recombine instances of good schemata, which results in producing new, equal- or higher-quality schemata. While the Schema theorem focuses on a single BB, it is important to consider the probability of different BBs mixing together, in order to innovate and recombine the best blocks [49]. The sampled data has to be representative, i.e., schemata have to be sampled at different points of the search space. Crossover does exactly that, constantly mixing schemata in different ways, thus supplying representative data for the sampling process.

The relationship between selection and crossover can be interpreted as the race between the takeover time and the innovation time [49, 50, 53]. The takeover time t^* is defined as

the time that it takes to increase the number of copies of the superior individual from one to $(n - 1)$, in the population of n individuals. The takeover time is related to the parameter known as the selection (or selective) pressure. If the takeover time is large, the selection pressure is low, because it takes a long time for the fittest candidate to take over the population. The innovation time t_i is defined as the time it takes crossover or another innovation operator to produce a solution that is better than any solution found to date. If the innovation time t_i leads, i.e., $t_i \leq t^*$, then innovation operators will create a better individual before the current best individual takes over and dominates the population. After that, the newly created best individual starts to work its way to domination, and the innovation clock resets. The condition $t_i \leq t^*$ is the most advantageous for the GAs success, because it creates the steady-state innovation. If the takeover time t^* leads, i.e., $t^* \leq t_i$, the current superior individual continuously increases its share, and eventually takes over the population. By the time t_i when the innovation operator might insert a better individual, it is too late: the diversity of the population is lost, and the latter becomes flat.

In the classical theoretical framework, the Simple Genetic Algorithm (SGA) is considered a special case of the Random Heuristic Search (RHS) [170, 171]. RHS is defined in terms of a search space Ω of cardinality (dimensionality) n, a collection of elements P (population) of cardinality r (population size), and a transition rule τ producing P_{i+1} from P_i. A simplex Λ is defined such that vector $\boldsymbol{p}$ belonging to Λ corresponds to a population P, and $p(i)$ represents the fraction of the element i (the probability of i) in the population. A heuristic function G: $\Lambda \rightarrow \Lambda$ is introduced that

produces the vector of the probabilities of the elements of Ω being selected, i.e., G specifies the sampling distribution for generating the next population. Given the current population vector $\boldsymbol{p}$, the next population vector can be obtained as $\boldsymbol{q} = G(\boldsymbol{p})$. Then RHS can be represented as the iteration over G:

$$\boldsymbol{p},\ G(\boldsymbol{p}),\ G^2(\boldsymbol{p}),\ldots\ . \tag{3.2}$$

The instance of the RHS process is called "focused" if G is continuously differentiable, and the sequence (2) converges. The corresponding operator G is also called focused. The points $\boldsymbol{\omega}$ satisfying $G(\boldsymbol{\omega}) = \boldsymbol{\omega}$, are called the "fixed points" of G. The actual expression for G is derived using the theory of Markov chains [170]. If the mutation rate p_m is nonzero, all states of the Markov chain have a nonzero probability of being reached, which makes Markov chain ergodic. The ergodic Markov chain has a limiting, steady-state distribution, corresponding to the global optimum of the problem. Moreover, the probability of reaching the steady-state distribution does not depend on the initial state of the algorithm, i.e., on the initial population. The important outcome of the classical mathematical model is the rigorous theoretical proof that the Simple Genetic Algorithm asymptotically converges to the global optimum solution of the optimization problem, starting with any randomly seeded initial population.

Important classical results obtained in 1970s and 1980s laid out the EAs foundation, and influenced the following research in the theory and practice of Evolutionary algorithms. During the past 10-15 years, a number of new fundamental EAs concepts have been developed. They show that some of the earlier obtained theoretical results are incomplete,

or even incorrect. In particular, the following classical results have to be re-evaluated [36, 38]:

1. The belief that it would be possible to design a universally superior evolutionary search algorithm, as well as universal settings that would perform equally well on all problems.
2. The concept of maximization of implicit processing of different contending schemata, under a low-cardinality alphabet (e.g., binary representation).
3. The usefulness of the Schema theorem in predicting the propagation of schemata under noisy conditions, when fitness is a random variable.
4. The optimality of fitness-proportional selection.
5. The uniform improvement of Evolutionary algorithms, when crossover is used in addition to mutation.

Considerable effort was spent in the evolutionary community on the attempt to find the best set of parameters and operators of Evolutionary algorithms, with most of the research conducted on specific sets of test functions and benchmarks. However, the No Free Lunch Theorem essentially says that the conclusions based on sample results can be strictly applied only to those test functions and benchmarks for which they were obtained [175]. Attempts to find the best universal population size, the best crossover or mutation rate, etc. are pointless, without the reference to a particular class of problems to which they can be applied. In order for an algorithm to perform better than a simple random search, it has to reflect a particular structure of the problem. But the specific match will inevitably lead to a mismatch with the structure of some other problems. Therefore, the

scope of the algorithm has to be restricted to the data that reveal the advantageous structure.

While the classical works emphasize the importance of low-cardinality schemata and binary representation, it was shown that the equivalent Evolutionary algorithm can be deduced for any cardinality or representation [36, 38]. Moreover, there is no particular advantage in using any specific type of a single- or a two-parent genetic operator (e.g., crossover), since another respective single- or two-parent operator can be designed that will display the equivalent performance, for any other type of representation.

The Schema theorem was derived under the implicit assumption of deterministic values of the contending schemata. In most practical scenarios, however, the evaluation of fitness includes some random effects, e.g., computational or measurement errors. The observed fitness is described by random variables that have some probability distribution. Randomness introduces bias: the expected sampling from different schemata will no longer be proportional to the ratio of the schema mean value, to the sum of the means of all competing schemata. The Schema theorem cannot adequately describe the mean sampling of the alternative schemata, when their fitness values are random variables.

Fitness-proportional selection scheme is not optimal, as it has the potential to lose some of the best solutions over iterations. Instead, the elitist evolutionary scheme should be used in practice. Moreover, more versatile variation operators have to be used, like multi-point, and particularly uniform crossover, which outperform single-point crossover on

many problems. This can be attributed to the fact that uniform crossover can generate offspring in the entire search space, while single-point crossover is limited to only a subset of potential solutions. At the same time, a balance has to be maintained between the exploration of new sub-spaces (which uniform crossover does) and the exploitation of already gained positive results. The effect of crossover has to be studied in conjunction with other EAs parameters, such as population size, mutation rate, etc.

Recent theoretical and empirical studies of the relative value of crossover and mutation show that traditional view of crossover as the major engine behind EAs, with mutation playing a background role, is not justified. In many cases, the presence of crossover does not significantly improve the search that uses mutation alone. The effect of crossover can vary depending on the problem at hand, the landscape of the objective function, and the relationship with other EAs parameters. Crossover is particularly useful at the beginning of the search, because it provides variations with larger step size. Since the initial random population most likely is located away from the optimum, crossover helps solutions depart more quickly from the initial set of points.

Although important foundational results have been obtained using Markov chains, it is difficult to capture the entire EAs dynamics within that framework, which attempts to describe the high-dimensional, highly nonlinear, random behavior of EAs on a micro-level. One of the more promising approaches is based on the macro-behavior of the algorithm; it operates with just a few integrated parameters, and utilizes the apparatus of statistical mechanics [11]. The main parameter of the algorithm is the fitness distribution,

described in terms of the cumulants around Gaussian distribution. A model predicting the long-term evolution of the population can be built on the basis of evolution of the cumulants during one generation.

3.2. Implementation of Evolutionary Algorithms

Numerous variants and models of Evolutionary algorithms have been developed, that differ in the details of the EAs implementation. This section starts with an overview of the main representation schemes and some of the issues related to the evaluation of the quality of a candidate solution. Thereafter, commonly used modifications of evolutionary operators - selection, crossover, and mutation, - are discussed. The section finishes with the review of the main EAs control parameters and mechanisms, and discussion of their effect on the performance of the algorithms.

3.2.1. Representation of Candidate Solutions

Representation of the EAs candidate solution to optimization problem maps the space of feasible solutions onto the space of their encoded counterparts that have specified data structures. The encoding schemes currently used in EAs [44] fall under the following main categories:

- classical binary encoding;
- real-valued encoding, best suited for functional optimization problems;
- integer or literal permutation encoding, best suited for combinatorial optimization problems;
- general data structure encoding.

From the standpoint of the structure, one-dimensional chromosomes are most commonly used; although, in some combinatorial problems (e.g., transportation and scheduling) multidimensional chromosomes can be found. The encoded content of a chromosome can include only object parameters, i.e., the vector of the desired solution to the problem, or the vector of the solution, with some additional EAs parameters (e.g., population size, mutation rate, etc.). The latter is routinely used in Evolution strategies, in order to facilitate the evolutionary self-adaptation of the control parameters, by applying evolutionary operators not only to candidate solutions, but to EAs parameters as well.

Historically, the earlier works on Genetic algorithms focused on binary encoding, i.e., on representing candidate solutions as binary strings. The encoding still remains one of the common approaches, particularly in combinatorial optimization problems. In classical binary encoding, a bit string typically has a fixed length and a fixed order. Much of the theoretical work and heuristics concerning EAs parameter settings is based on this type of encoding.

Evolutionary strategies from the very beginning have been working with real-valued parameter encoding. In many practical applications, binary encoding seems unnatural and can make the EAs formulation of the problem significantly and unnecessary complex, and even lead to a poorer performance. It was analytically proved that within the class of bijective mappings, no one choice of representation offers a unique advantage [39]. Therefore, the recommended best practice is to pick representation that is intuitively most appropriate for the problem at hand [109].

Apart from the popular binary and real-valued encoding, some other interesting encoding schemes have been developed and successfully used in different application areas. For example, grammatical, multi-character encoding is used to design evolving neural networks [85]. Genetic programming is used to evolve Lisp programs in [88, 89]. The idea is based on the fact that Lisp program can be represented as a parse tree. Consequently, a candidate solution is encoded as a tree consisting of functions and terminals. In GAs with stochastic encoding, every candidate solution is binary encoded using multivariate Gaussian distribution, with the mean vector μ and the variance matrix Σ [91]. A given binary string, therefore, represents a region of the search space, rather than just one point, as in the conventional representation scheme.

3.2.2. Fitness Evaluation Function

Fitness evaluation function provides the principal link between Evolutionary algorithms and a particular optimization problem to be solved. The function evaluates the quality of a candidate solution encoded in a chromosome, based on the objective of the search. Higher-quality individuals usually have a higher probability of survival and, therefore, a higher probability of selection for the reproduction pool. It is critical to carefully design the evaluation function in such a way that it can capture the essential properties of the problem. In optimization, fitness is usually evaluated as the objective function $f(x)$, or some kind of its surrogate, e.g., a scaled fitness value.

One of the challenging problems in the application of Evolutionary algorithms is the control of the selection pressure. The latter should be lower at the beginning of the

search, in order to avoid the premature convergence. It should gradually increase toward the end of the search, in order to emphasize the quality of the most successful survived individuals. Different scaling mechanisms have been introduced allowing for the control of the selection pressure during the algorithm run [107]. One of the popular mechanisms, sigma scaling is an adaptive technique that incorporates problem dependent information into the evaluation function as follows:

$$f'(x) = f(x) + (f_{avg}(t) - c\sigma), \tag{3.3}$$

where $f_{avg}(t)$ is the average fitness of the population at the time t, σ is the standard deviation of the population fitness, and c is a constant, usually a small integer number [107].

3.2.3. Evolutionary Operators

A variety of evolutionary operators have been suggested in the research literature. Some operators are unary, and create new individuals by a specified change in a single individual (e.g., mutation and inversion). Others are higher-order operators creating new individuals by combining parts of two or more parents (e.g., various types of crossover). The choice of operators depends on many factors, including the specifics of the problem and representation of a candidate solution. Feedback provided by the fitness distribution can be used to identify the best choice of operators, as well as EAs parameters and their settings. Most commonly used evolutionary operators of selection, crossover, and mutation are briefly discussed in this section, including their different types and modifications.

Selection in Evolutionary algorithms is responsible for choosing parents that will participate in producing offspring for the next generation of chromosomes. Thus, the selection mechanism plays a key role in the algorithms guiding the search in the direction toward the fitter fraction of the population. All selection mechanisms are aimed at creating more copies of higher-fit chromosomes, but differ in the techniques used to allocate copies to the fitter individuals. The main problem during selection is reaching a balance between the temptation to operate only on the best fraction of the population, and the need to maintain sufficient population diversity. In the EAs community, this problem is known as exploitation of the best qualities obtained to date, versus exploration of the search space [4].

The classical stochastic sampling procedure known as fitness-proportional (or fitness-proportionate) selection requires that the expected number of times $ExpVal(s)$ a chromosome is selected for mating is proportional to its fitness, divided by the average fitness of the population [66]. Solution with higher fitness values most likely will receive more copies than solutions with smaller fitness values. In practice, the population size is often small, and the number of trials obtained from the algorithm can significantly differ from its expected value, due to the sampling errors. The variant of the fitness-proportional selection, the Stochastic Universal Sampling (SUS) minimizes the difference between the expected and the actual number of trials [7].

Rank-based (or ranking) selection attempts to prevent the premature convergence of the algorithm, and achieve a better leverage of the selection pressure [6]. The expected

selection value of every individual depends on its rank, rather than on the absolute fitness value. The ranking mechanism leverages the selection pressure: it distributes the expected selection values more evenly across the population at the beginning of the search, when the fitness variance is high. As the search progresses, and the fitness variance decreases, the same mechanism still maintains the relative ratio of the expected number of trials between the neighbors in the rank list.

Tournament selection is similar to rank-based selection, from the standpoint of selection pressure, but more efficient computationally [50]. It does not require time-consuming sorting of individuals according to their fitness. Stochastic binary tournament selection picks at random two candidates from the population, and a random number r between 0 and 1. Number r is compared with a pre-set parameter k ranging between 0 and 1. If $r < k$, the fitter candidate is selected to be a parent; otherwise, the less fit candidate is selected.

Different strategies can be used to form a new generation, depending on what fraction of the parental population and that of the offspring population are retained. Elitist approach requires that some number E of the best individuals from every generation remains intact, as opposed to generational approach that requires a complete replacement of parents with the offspring [24]. Elitism ensures that the best qualities accumulated by the EAs process to date will not be lost during crossover and mutation, as evolutionary search progresses.

The operator of selection does not create new individuals; it only helps promote good solutions; creation of new individuals is the responsibility of recombination. Crossover

has been traditionally considered the most powerful recombination operator in Genetic algorithms. It is also the primary operator that distinguishes Genetic algorithms from other stochastic search methods. Different types of crossover have been proposed in the research literature, but no single recommendation for all problems can be educed from the research studies. Crossover attributes, like loci bias, disruptiveness, and its ability to produce or combine many schemata in just one step, depend on the specifics of the problem, as well as on the details of the EAs implementation (e.g., population size, or selection pressure).

Traditionally, single-point crossover has been the simplest, yet the most commonly used form of crossover, in both, binary and real-valued encoding [66]. Given two parents, a single crossover position on the chromosome is chosen at random, and parents exchange their substrings after the crossover position, thus forming two offspring. In multi-point crossover, a few breakup positions can be chosen completely at random, or according to some probability distribution. By and large, the multi-point operator can produce a larger variety of schemata than the single-point operator does. In the uniform crossover, every bit of a binary string can potentially be a crossover point [150, 153]. A string of bits, called a crossover mask, of the length of the chromosome, is generated at random, with the probability 0.5 for each bit. The 1-bits in the mask correspond to the parental bits subject to exchange. The parity of each mask bit indicates from which parent the offspring will receive the corresponding bit. Uniform crossover can be easily extended to real-valued encoding, if crossover points are placed between the float-point numbers on the chromosome corresponding to the encoded parameters.

Apart from binary crossover that exchanges information between binary-coded parents, a number of recombination operators have been proposed for real-valued encoding, which average, or blend the values across the parents [32, 110]. Although, in nature, new individuals are created via either asexual or sexual reproduction, a few operators have been proposed that use panmictic (multi-parent) crossover, e.g., a gene scanning operator produces one offspring from $n > 1$ parents [31].

While the best choice of crossover operator is problem-dependent, the important property of this operator, its mixing capability, can be estimated independently of the problem. The mixing capability determines how quickly the population can deviate from the initial random distribution, in order to explore the entire search space. It also determines disruptive, as well as recombination power of crossover, i.e., the rate of combining different schemata together in one genotype. Analytical and experimental studies conducted on a generalized card-shuffling problem show that uniform crossover mixes the population the fastest (the shuffling time $\tau = 1.44$), followed by two-point crossover ($\tau = 0.5L$), then by single-point crossover ($\tau = L - 1$), where L is the number of cards in each of N suits [125]. Consequently, multi-point crossover with n crossing points chosen at random would fit between the two-point, and the uniform operators.

Unlike crossover, mutation is an asexual operator acting upon a single chromosome. Mutation has always played the dominant role in Evolutionary programming and Evolution strategies, while in Genetic algorithms it was considered a secondary, or background operator. The recent tendency is to recognize the role of mutation as a search

tool, and as an important instrument for retaining the diversity of the population [111]. When the algorithm starts converging to some promising solution area, the number of chromosomes representing this area grows fast. In the finite population, the current successful individuals crowd out their less successful counterparts, thereby eliminating the alleles that the latter carry. Since crossover does not produce new alleles, the eliminated alleles can be lost altogether. One cannot be sure, however, that the lost alleles do not belong to some better solution. Under these circumstances, mutation is the only classical mechanism that can recover the lost alleles, and bring them back into the gene pool.

In a binary string, the commonly used bit-wise mutation is implemented by flipping a bit from 0 to 1, or vice versa, with the mutation probability p_m [66]. The simplest mutation scheme for the real-valued encoding is to pick an offspring $O(x')$ from the search space $[x_L, x_U]$ at random, as follows:

$$x' = r\,[x_U - x_L], \tag{3.4}$$

where x_U and x_L are the respective upper and lower bounds of the search space, and $r \in [0,1]$ is a random number [107]. Another common way is to relate mutants to their parents, like in the non-uniform adaptive mutation scheme [107], or in Evolution strategies [4, 144].

3.2.4. Control Parameters of Evolutionary Algorithms

The right choice of control parameters of Evolutionary algorithms significantly affects the computational performance and the overall success of the evolutionary search.

Control parameters most noticeably include population size and probabilities (rates) of crossover and mutation. Few theoretical results for relatively simple optimization problems have been obtained in this area; and most recommendations are based on experimental studies conducted on certain classes of test problems.

Most research, including multi-population models, is focused on finding the optimal population or sub-population size, before the algorithm run. One of the earlier models singles out the best building block, and considers fitness contributions from other partitions as noise [52]. The model assumes that if wrong blocks were selected in the first generation, the algorithm would never recover, and would converge to a false solution. On the other hand, the correct initial selection would lead to the correct optimal result. The population size n then relates to the quality of the decision, according to Formula

$$n = 2c(\alpha)2^k m' \frac{\sigma_M^2}{d^2}, \tag{3.5}$$

where α is the probability of failure, $c(\alpha)$ is the square of the unit normal distribution for the probability of failure α, k is the order of the building block, m' is one less than the number of building blocks in a string, σ^2_M is the average fitness variance, and d is the fitness difference between the best and the second best blocks.

For a long time, the fixed optimal settings $p_c \approx 0.6$ and $p_m \approx 0.001$ for crossover and mutation rates were common in the GAs community. The rates were obtained on a specific test suite for binary encoding, and served as the basis for numerous studies and applications [24]. Two measures were devised to evaluate the performance of the algorithm. Offline performance measures the overall convergence, i.e., the quality of the

best solution obtained at the end of the algorithm run. Online performance is evaluated as the average result obtained to date over all generations, up to, and including the current generation.

Based on the analysis of the race between selection and crossover [49], in terms of the takeover time t^* vs. the innovation time t_i, the lower estimate of the crossover was derived as

$$p_c \geq \frac{\ln(s)}{p_s N \ln(N)}, \tag{3.6}$$

where s is the selection pressure, p_s is the probability of success during a single crossover operation, and N is the population size. The minimum crossover probability grows proportionally to the logarithm of the selection pressure. The actual probability has to be greater than the minimum value, in order to satisfy the steady-state innovation condition of the algorithm success.

The variety of the EAs parameters and the complexity of their nonlinear interaction make the task of manual tuning the optimal settings for a particular problem very difficult. Significant efforts have been made to find various adaptive schemes of control [21, 64, 109]. Utilizing self-adaptation, for example, gives an opportunity to customize the EAs settings concurrently with conducting the search for the optimum solution. The initial optimization problem can be re-formulated then as the concurrent optimization search in two different spaces: the space of all possible solutions to the original problem, and the space of all possible Evolutionary algorithms (together with their parameters) that can be applied to solve the original optimization problem.

3.2.5. Maintaining Population Diversity and Mating Restrictions

One of the strong appeals of the evolutionary approach stems from the concept of maintaining a population of potential solutions, as opposed to a single solution. The population typically samples the entire search space at the beginning of the search, thus allowing for exploration of different search sub-spaces. As the selection pressure marks out the fittest candidates, the search more and more focuses on a few sub-areas, thus decreasing the diversity of the population. In the case of a multimodal objective function having multiple optimum peaks, the population size and selection pressure are often not adequate to the fitness landscape. The inadequacy can have the negative effect of overlooking some of the potential optimum peaks, especially at the beginning of the search. Different approaches, including niching and crowding, have been developed, in order to help maintain the population diversity at a satisfactory level.

Niching penalizes solutions that are close to each other, and forces the population to spread across the multiple extreme areas, in such a way that the number of individuals in each area is proportional to its height [48, 51]. Niching requires that all individuals within the same region of the search space share their fitness, so the individual fitness becomes

$$f_i' = \frac{f_i}{\sum_{j=1}^{N} sh(d(i,j))}, \tag{3.7}$$

where N is the population size, f_i is the regular fitness value, $d(i, j)$ is the distance between the individuals i and j, and $sh(\cdot)$ is the sharing function defined as follows:

$$sh(d) = 1 - (d / \sigma_{\text{share}})^{\alpha}, \text{ if } d < \sigma_{\text{share}}, \text{ and } sh(d) = 0, \text{ otherwise.} \tag{3.8}$$

Here, σ_{share} defines the size of the neighborhood around i (niche radius), and α is the scaling factor. The sharing function determines the degradation of the individual's fitness, due to crowding by the neighbors.

Crowding is another strategy that simply replaces old individuals with new ones, when they are similar, thus weakening the tendency to accumulate similar solutions [24, 27]. Along with maintaining the population diversity and proper allocating chromosomes in the search space, another concern in multimodal function optimization is crossing over solutions belonging to different optimum peaks. If the regular crossover was applied uniformly to the entire population, some search efforts will be wasted on recombining solutions belonging to different optima, thus obtaining lethal offspring. In order to maintain the production of viable offspring, different speciation methods have been designed that restrict mating patterns [27].

3.2.6. Terminating Criteria

The termination condition is "one of the fuzzy parts" of Evolutionary algorithms [61]. The following criteria are commonly used in the EAs practice:

1. The quality of the best chromosome is checked, and if it is an acceptable solution to the problem, the search is terminated. This method requires a good understanding of the specifics of the problem, as well as the knowledge of the possible range of acceptable solutions.
2. The simplest and frequently used way to terminate the evolutionary process is to limit the number of generations, or, alternatively, the number of fitness function

evaluations. This method requires some prior experimental or theoretical knowledge of the algorithm's behavior for a particular class of problems.

3. More meaningful approach is to terminate the search, when no significant improvement of the result can be achieved. There are two main techniques that are used down this line. The first technique measures the progress achieved in relation to the chromosome structure (genotype). It checks the number of converged alleles within the chromosomes. Allele is considered converged if a certain pre-set fraction of the population has the same allele value. The second technique measures the progress achieved on the chromosome level (phenotype). If the fitness improvement over some pre-set number of generations is small, the process terminates. The fitness improvement can be monitored, in relation to the best chromosome in the population. Alternatively, the mean and the standard deviation of the fitness of the entire population can be measured.

3.3. Advanced Models of Evolutionary Algorithms

Evolutionary algorithms have proved to be a versatile and flexible approach, well suited for solving complex optimization problems. The versatility comes from the fact that EAs in their purest classical form implement an adaptive random search that does not employ any domain-specific knowledge. It can be easily applied to the entirely different problem spaces. The paradox is that the versatility of the classical EA is, at the same time, its main bottleneck. Many ad hoc methods that use domain-specific knowledge relevant to a particular class of problems might likely outperform the classical Evolutionary algorithm on those problems. Here is where the flexibility of EAs comes to the rescue: the classical

model can be easily modified and extended, to include virtually any domain-specific knowledge, or combined and enhanced with other, more traditional algorithms and methods. This section gives a brief overview of some interesting advanced EAs models stemming from the practical needs to solve various real-world problems.

3.3.1. Hybrid Evolutionary Algorithm

Hybrid Evolutionary Algorithm (HEA) enhances global evolutionary search with local search based on traditional local optimization techniques [2, 60]. There are several ways of incorporating local search into the global evolutionary procedure [80]:

1. Run the standard (i.e., classical) Evolutionary algorithm, and then apply local search to refine the final optimal solution obtained with EA.
2. Use local search as an evaluation function for the EAs search. In that case, EAs generate the starting points for local improvement; and thereafter local search evaluates the fitness of the candidate solutions, by investigating their local neighborhoods. In some cases, this approach can give rise to a problem of multiple genotype-to-phenotype mapping, when different individuals map to the same local optima.
3. Use local search as a genetic operator, in a way similar to mutation. Local search is applied only to a selected small fraction of the population, in order to reduce the computational cost. This approach can be disruptive for the schemata processing, because it alters the implicit information about the hyperplane partitioning of the population.

4. Mix the previous strategies, by updating the genotype of only a fraction of the population.

In order to emphasize the role of local search in EAs, the term "memetic algorithm" is introduced in [112], and a precise formal description of memetic algorithm is given in [126]. According to the description, memetic algorithm is a GA, in which a local optimizer is applied to every offspring, before it is inserted into the population. The operators of recombination can produce solutions that are outside the subspace of local optima, in which case the local optimizer repairs the solutions, and produces the final population that fits within the optimal subspace. Memetic algorithm, therefore, can be thought of as a kind of EA over the subspace of local optima.

Genetic algorithm is enhanced with the combination of the neighborhood search (i.e., local search), and a technique called "path tracing", in [132]. Both are embedded into the general evolutionary framework. Path tracing plays a role of an "intelligent" form of recombination, in place of the regular operators of crossover and mutation. In the path tracing, crossover acts as interpolation between two parents, and mutation acts as extrapolation from one of the parents. The algorithm makes a probabilistic choice between the neighborhood search and path tracing. With the probability P_x, either interpolating, or extrapolating path tracing is used, otherwise local search is used, around the first selected parent. In order to reduce the computational cost of local search, the latter is made stochastic, similar to Simulated annealing, with the fixed number of iterations.

3.3.2. Messy Genetic Algorithms

The Fast Messy GA (FMGA) emphasizes the role of crossover, as the main instrument of innovation that is vital for the GA to succeed [54]. The classical Genetic algorithm does not investigate the linkage between different genes in a candidate solution. There are two types of linkage that occur in a chromosome: genetic and epistatic. Genetic linkage is defined as the proximity of genes, and their tendency to be transferred together during the crossover operation. More important epistatic linkage is defined as interdependency of different genes, regardless of their position on a chromosome. The Fast Messy GA explores epistatic linkage, and differs from the classical GA in the following way:

- using messy solution encoding;
- using messy genetic operators;
- breaking the search process into three heterogeneous phases;
- using level-wise iteration, to improve the complexity of the solution.

FMGA allows for the individual's genotype to be either under-specified, or over-specified, e.g., the string ((4, 0) (1, 1) (2, 0) (4, 1) (4, 1) (5, 1)) is under-specified, with respect to gene 3 (no instances), and over-specified, with respect to gene 4 (3 instances). The operator of messy crossover is broken down into two operators, cut and splice. The cut operator divides the string into two parts, while the splice operator joins the head of one string with the tail of another string, to form the offspring. The resulting offspring can have different lengths.

The Gene Expression Messy GA is a variant of the Messy GA that uses a transcription operator [83]. Transcription results in identifying and storing good-linkage gene subsets. These subsets are then processed at the juxtapositional phase, by means of selection and recombination. The Linkage Learning GA replaces deterministic gene expression in the Messy GA, with probabilistic gene expression [58]. Probabilistic expression begins at a random locus, and moves in a clockwise fashion. Alleles are expressed probabilistically, depending on the starting locus and the relative distance between the alternative alleles.

3.3.3. Hierarchical Genetic Algorithm

A model of the Hierarchical Genetic Algorithm (HGA) is based on the observation that genes in a chromosome of a living organism are arranged in a hierarchical manner [101]. Some genes are dominating the others; and there are active and inactive genes. In a chromosome, genes can be classified as regulatory sequences (RS), and structural genes (SG). While structural genes serve as the building blocks of proteins, the regulatory sequences serve as the leaders that identify the beginning and the ending of the SGs, participate in turning on and off transcription of SGs, and function as initiation points for replication or recombination. The Hierarchical Genetic Algorithm attempts to mimic the biological DNA structure, with the application to an artificial environment. The HGA chromosome consists of parametric genes (analogy of SCs in the DNA) and control genes (analogy of RSs in the DNA). Control genes are organized in a hierarchical manner. The activation of a parametric gene is controlled by the value of the 1st level control gene, which, in turn, is controlled by the value of the 2nd level control gene, etc. Unlike the classical GA that has fixed chromosome and phenotype structures, HGA operates without

these constraints, and a content of varying length is available within the same chromosome structure. HGA searches concurrently for the solution to the problem, and for the optimal hierarchical chromosome structure, so that the final optimization objective can be met.

3.3.4. Implicit Redundant Representation Genetic Algorithm

Interesting variant of Evolutionary algorithms, called Implicit Redundant Representation Genetic Algorithm (IRRGA) is developed in [128]. The algorithm can change "on the fly" the number of optimization variables, i.e., the dimensionality of the search space. It does so by making the number of optimization variables also a variable that has to be optimized during the main search for the optimum solution. IRRGA uses redundancy in the solution representation, which provides a two-fold benefit. First, redundant segments act like introns in a chromosome of a living organism, and help protect the active variables from disruption, during the operations of crossover and mutation. Second, redundant segments serve as potential placeholders allowing for adding new variables dynamically, during the algorithm execution.

3.3.5. Virus-Evolutionary Genetic Algorithm

The classical Evolutionary algorithm is rooted in Neo-Darwinian theory of evolution stating that evolution changes living organisms, in the process of natural selection and heredity. Other evolutionary theories have been proposed, based on the progress of molecular biology. One of them is the Virus theory of evolution stating that the key mechanism of transporting DNA segments across the species is virus transduction, i.e.,

genetic modification of a bacterium by genes from another bacterium [42]. Virus infection is capable of rapid horizontal propagation throughout the population, while heredity allows for genetic information to propagate vertically from generation to generation. The Virus evolution theory can explain the punctuational evolution in nature. A new variant of Evolutionary algorithms called VEGA (Virus-Evolutionary Genetic Algorithm) was developed based on the Virus evolutionary theory [42]. There are two types of population in VEGA: the host population of candidate solutions, and the virus population. The virus population has two specific virus infection operators aimed at transmission of the DNA segments among the host individuals. The first operator, a reverse transcription, transcribes virus genes onto the host chromosome. The second virus operator, transduction, creates a new virus, by transducing a DNA segment from the host chromosome.

3.3.6. Cultural and Agent-Based Algorithms

A number of EAs variants attempt to enhance the original biological framework with the content drawn from social behavior of the living organisms. In cultural algorithms, individual solutions learn during their lifetime, both on genetic and social level [133]. Cultural algorithms include population structure, belief structure, and communication channel. Individuals inherit properties at two levels: at the micro-evolutionary level they inherit the behavioral and genetic traits, and at the macro-evolutionary level they inherit the beliefs of the population. The two levels interact through a communication channel allowing for mutual adjustment and influence of the both evolutionary levels.

Agent-based models in evolutionary computations were introduced in the 1990s. A potential solution to the problem can be viewed as an agent whose behavior is based on some set of rules applied at the micro- or macro-level. In general, the rules can be adaptive, and tailored to a particular problem. For example, in the Patchwork model, every individual is modeled as a mobile agent that has a number of properties including the maximum life span, the ability to breed, the fitness-based mortality, and the ability to have preferences in decision making [90]. Agents move and interact in a 2-D grid of patches (cells) in such a way, that they can collect information only from local environment, and can affect only that environment. A set of adaptive behavioral rules regulates the agents' lives.

3.3.7. Using Statistical Analysis in Evolutionary Computations

In spite of the presence of some theoretical grounding for Evolutionary algorithms, they remain, to a large extent, experimental methods, when it comes to real-world problems. Since they are inherently non-deterministic, it seems logical to incorporate well-known robust statistical techniques into the EAs framework, in order to gain a better understanding of the search process, and to augment its heuristic with an intelligent analysis.

Statistical analysis tools can be employed for the dynamic run control, data preprocessing, and estimating the statistical significance of the results [8]. Although [8] refers to Genetic programming, the findings are generally applicable to other variants of Evolutionary algorithms. Considering GP as an unsupervised learning procedure, the

investigators outline the following aspects of statistical analysis that can be effectively employed in the evolutionary search:

1. Offline pre-processing and analysis helps determine which data instances should be input into the GP system.
2. Offline post-processing measures the computational effort, i.e., the number of candidate solutions that have to be processed, in order to find the optimal solution.
3. Online analysis and measurement of the data monitor transitions between randomness and stability in the interim results, during the algorithm run. They also increase the probability of the intelligent control of the algorithm performance on the fly, via feedback from the online data analysis.

A probability model for estimating the number of local optima (attractors) in the fitness landscape using the data from repeated sampling is presented in [131]. Such estimation can be helpful in assuring the quality of the solution obtained by the evolutionary search, and in estimating the computational cost of finding the global solution. Although the estimates derived in [131] are based on the assumption of isotropically distributed attractors (i.e., uniformly distributed in the fitness landscape, and having nearly equal basin sizes), the paper points out that the estimates can still be used to compare the effect of different recombination operators.

3.3.8. Estimation of Distribution

Estimation of Distribution Algorithm (EDA) is a relatively new paradigm, and one of the fastest growing techniques in evolutionary computations [94]. Although EDA is rooted in Evolutionary algorithms, it goes far beyond the traditional genetic operations of random search, and replaces crossover and mutation with learning and sampling the probability distribution of the best individual solutions in the population. In this way, EDA attempts to capture relationship between the variables, in order to build a distributed model of good solutions across the population, and to exploit this information in a competent way, thus more efficiently guiding the search toward the most promising areas of the search space. While in the classical Evolutionary algorithm, the relationships between building blocks are not expressed explicitly, in the Estimation of Distribution Algorithm, these relationships take an explicit form of the joint probability distribution between individual candidate solutions selected at every iteration.

The key problem in EDA is estimating the probability distribution. Computing all parameters needed for the precise estimate is intractable, and a number of approximations have been proposed that factorize the distribution, according to a chosen probability model. The Univariate Marginal Distribution Algorithm uses the simplest model, and estimates the joint probability distribution, as the product of independent univariate marginal distributions [114]. The Compact Genetic Algorithm is a variant of EDA that works for a binary solution representation; it initializes the vector of probabilities following Bernoulli distribution [59]. The Mutual Information Maximization Algorithm looks at the statistical dependencies between the pairs of variables (pair-wise probability

distribution) [22]. The Bayesian Optimization Algorithm employs an acyclic Bayesian network, in order to construct and explore the probabilistic model of the best individuals in the population; and to identify and exploit the gene linkage [120].

3.3.9. Parallel and Multiple-Population Evolutionary Algorithms

Evolutionary algorithms are inherently well suited for parallel processing, due to the fact that they concurrently work with the population of independent solutions. The population can be distributed among several processors, which would significantly speed up the computations. Many efforts have been made to design efficient parallel versions of EAs. Despite the natural distributed character of EAs, their parallel implementation, however, is a complex nonlinear problem controlled by many parameters that require careful consideration and tuning. The choices of topologies and migration rates; the number and the size of subpopulations are all closely related, and have to be considered simultaneously. Four major types of the algorithm are identified in [16] as follows:

- single-population master-slave EA;
- multiple-population EA;
- fine-grained EA;
- hierarchical hybrids.

Master-slave EA has a single population. The master node stores the population, executes genetic operators, and distributes the population among the slaves. The slaves evaluate the fitness of the individual candidates, and return the results back to the master node. This is the simplest form of the parallel EA, also known as the *global* parallel EA.

Multiple-population, or multiple-deme EA consists of several subpopulations distributed over different processors. This class of methods is also known as *distributed EA*, *coarse-grained EA*, or *island model*. The subpopulations evolve independently, but exchange individuals, from time to time. This is one of the most complex models, because user has to decide on the number and the size of the subpopulations (demes); the frequency of exchange (migration); and the number, selection method, and destination of the migrants. Some researchers argue that this model is a closer analogue to the natural population that the single-population EA model. The model is particularly well suited for MIMD (Multiple-Instruction-Multiple-Data) computers with distributed memory.

Fine-grained EA has a single spatially structured population. The structure is usually a rectangular two-dimensional grid, with one individual per node. The evaluation of fitness function is performed simultaneously for all nodes. Selection and mating are restricted to a small neighborhood around an individual. Since the neighborhoods overlap, good traits can eventually spread across the entire population. This model is also called a *diffusion model* or *cellular EA*, due to its resemblance of the random diffusion of particles in fluid, or cellular automata with stochastic transition rules. The model is particularly well suited for massively parallel SIMD (Single-Instruction-Multiple-Data) computers.

3.3.10. Evolutionary Algorithms in Multi-Objective Optimization

Multi-Objective Optimization Problem (MOOP) is formulated as finding maxima (or minima) of several objective functions, under a specified set of constraints:

$$\max \{ z_1 = f_1(\boldsymbol{x}), z_2 = f_2(\boldsymbol{x}), \ldots, z_q = f_q(\boldsymbol{x}) \}, \; g_i(\boldsymbol{x}) \leq 0 \;\text{ for } i = 1,\ldots, m, \qquad (3.9)$$

where $\boldsymbol{x} \in R^n$ is the vector of n decision variables, $f_i(\boldsymbol{x})$, $i = 1,\ldots, q$ are the objective functions, and $g_i(\boldsymbol{x})$, $j = 1,\ldots, m$ are the constraint functions forming the area of feasible solutions [44].

Unlike in single-objective optimization, there is no single solution in MOOP that is best, with respect to all objective functions, because of the conflict among the latter. A set of optimal solutions exists, which cannot be compared to each other. The main property of the solutions is that no improvement is possible in any of them, without degrading or sacrificing at least one other solution. These solutions are called non-dominated, or Pareto-optimal; they form a Pareto front. Non-dominated solutions can be discriminated only on the basis of expert knowledge of the problem, or on the preference set used by the decision maker (DM). The numerical solution to MOOP, therefore, has two goals simultaneously:

- find a set of solutions that are as close to Pareto front as possible;
- find a set of solutions that are as diverse as possible, i.e., are sparsely spaced along the Pareto front.

In real-world applications, usually only one solution has to be chosen from the set, in accordance with the DM preferences. These preferences give a certain priority order to non-comparable solutions, based on the DM judgment. The final solution chosen from the set is called the best-compromised solution.

Two main approaches to solving MOOP are identified in [44] as follows:

1. Preference-based approach attempts to find a compromised or preferred solution from the beginning, thus effectively reducing MOOP to a single-objective optimization problem. A well-defined set of preferences or priorities of the objective functions has to be in place, before the numerical procedure starts. The advantage of this approach is usually a significant reduction in the complexity of the problem.
2. Generating approach seeks the entire set of Pareto solutions or their approximations. Very often, DM does not know a priori about the exact trade-off among different objectives. It is desirable first to find the entire set of Pareto-optimal solutions, and then to choose one solution, using some higher-level knowledge or considerations. This approach gives the decision maker the overall perspective of all possible solutions that MOOP can offer, and provides the option to choose the solution that most suits some criteria of importance or acceptance.

The classical methods of solving MOOP have shortcomings [26] that limit their usefulness in real-world applications:

- only one of the set of Pareto-optimal solutions usually can be found, during one run of the algorithm;
- if the problem is non-convex, some algorithms cannot find all Pareto solutions;
- all classical algorithms rely on some problem-specific knowledge, available prior to the algorithm run.

Evolutionary algorithms are inherently well suited for MOOP, because they maintain the population of potential solutions. Therefore, EAs can find and preserve a set of multiple Pareto solutions simultaneously. When EAs are applied to multi-objective optimization, the important issue is the design of the mechanism by which fitness can be assigned to different solution candidates [44]. Vector-evaluated, Pareto ranking, and random-weighting mechanisms are used in the generating MOOP approach; while compromise-based, adaptive weighting, and goal programming have been designed for the preference-based approach.

An extension of the classical Evolutionary algorithm to multi-objective optimization, known as the Vector-Evaluated Genetic Algorithm (VEGA) was first proposed in [140]. In the algorithm, the selection step works in a loop that selects an appropriate fraction of the population, on the basis of each objective function. VEGA does not use scalar fitness function, so individuals do not have an explicit form of the overall fitness. After the selection, the entire population is shuffled, to obtain fair mating conditions between all fractions, before crossover and mutations are applied.

Pareto ranking fitness assignment, as a way to achieve the equal reproduction for all Pareto solutions, was first suggested in [48]. The population is ranked on the basis of non-dominated solutions, as opposed to the regular ranking procedure in a single-objective EAs. In Pareto tournament procedure, two solution candidates $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$; and a comparison set $\{C\}$ are picked at random from the population [67]. Each of the two

candidates is compared against each solution in the set $\{C\}$, with the following outcome based on the comparison results:

- if one candidate, e.g., $\boldsymbol{x}_1$ is dominated by the comparison set, but the other $\boldsymbol{x}_2$ is not, then the non-dominated candidate $\boldsymbol{x}_2$ is selected;
- if both candidates are either dominated or non-dominated, the winner is decided on using the sharing technique: the solution that has the least number of neighbors wins the contest.

The random-weight EA implements a varying search direction toward the Pareto front [115]. The weighted-sum objective function is defined as follows:

$$z = \sum_{k=1}^{q} w_k f_k(\boldsymbol{x}), \tag{3.110}$$

where the random weights w_k are computed using nonnegative random numbers r_k (k = 1,…, q) as follows:

$$w_k = \frac{r_k}{\sum_{j=1}^{q} r_j}. \tag{3.11}$$

The compromise-based approach avoids computing all Pareto solutions, and can be useful when obtaining all Pareto solutions is either computationally expensive, or of little interest to the decision maker [44]. The approach identifies solutions that are closest to the ideal optimal point, with relation to some measure of distance. The ideal optimal point is the optimal solution that presumably satisfies all the conflicting objectives, and therefore cannot be achieved in reality. Since the ideal point is not attainable in actuality,

the concept of a proxy ideal point is suggested, that replaces the ideal point. The proxy ideal point is the ideal point that corresponds to the current generation only; it is expected to gradually approximate the actual ideal point, as the evolutionary search progresses.

The adaptive weight approach attempts to adjust the evolutionary search toward the Pareto front [44]. A good choice of the weighting vector is not usually critical, since weights are re-adjusted during the evolutionary procedure. Two extreme points, the maximal and the minimal are defined at every generation, for every solution in the population; they define the minimal hyperparallelogram that contains all current solutions. The maximal extreme point will gradually move toward the approximation of the positive ideal point (in the maximization problem).

The goal programming approach establishes a specific numeric goal for each objective function, and seeks a solution that minimizes the weighted sum of the deviations of the objective functions from their respective goals [44, 45]. Nonlinear goal programming method solves the problems with the conflicting nonlinear objectives and constraints. User provides the target of achievement for each objective, and gives the priorities, according to which the goals have to be achieved. The algorithm attempts to find the optimal solution that satisfies as many goals as possible, in the order of their priorities.

3.3.11. Evolutionary Algorithms and Fuzzy Concepts

The last decade has seen a growing interest among researchers to the application of the concept of fuzziness in evolutionary computations. There are two different aspects of the relationship between Evolutionary algorithms and fuzzy concepts [63]:

- incorporating ideas of fuzzy logic into Evolutionary algorithms, including fuzzy representation, fuzzy operators, and fuzzy parameter control;
- application of Evolutionary algorithms to fuzzy environment, particularly in the area of fuzzy optimization and fuzzy control.

The classical EA does not take into consideration the developmental progress from the gene level to the phenotype level [110, 169]. The progress can be considered as a fuzzy multistage decision process, and can be described by Fuzzy Evolutionary Algorithm (FEA) [169]. In FEA, a gene can have the real number, rather than the binary one, i.e., a binary gene becomes a fuzzy gene. Two different genotypes can produce the same phenotype, i.e., in the fuzzy representation, the one-to-one correspondence between genotype and phenotype does not hold anymore. In order to dynamically control the parameters of Genetic algorithm, the use of the fuzzy logic controller (FLC) is proposed in [97, 98]. Since the terminating criteria in Evolutionary algorithms are not well defined, a fuzzy stop criterion mechanism (FSCM) is developed in [106], which has some advantages over the traditional terminating schemes. FSCM relies on the user-defined acceptable percentage optimal solution, called α-cut. Instead of the crisp optimum solution, the algorithm searches for a fuzzy goal using the α-cut as the cut-off parameter.

The algorithm also estimates belief and uncertainty measures of the found solution, thus giving the user some kind of confidence in the obtained results.

Finding the exact optimum solution in fuzzy optimization is meaningless for the decision maker. The latter wants to find a neighboring domain of the optimum solution, such that every solution within the domain can be "optimum" under the set of preferred criteria, in the environment characterized by the fuzziness of the objective function and constrained resources [44, 45]. The preferred solution can be identified by the DM via human-computer interaction. Such interactive method for maximizing the minimum satisfaction degree is proposed in [156]. Application of fuzziness in multi-objective 0-1, integer, and nonlinear programming with Genetic algorithm is also described in [137], where a modified interactive version of the GENOCOP III system is applied to solve fuzzy multi-objective nonlinear programming. A membership function $\mu_i(f_i(\boldsymbol{x}))$ for each of the objective function is defined, where the corresponding reference membership levels μ_i are elicited from the decision maker using human-computer interaction. Then, the corresponding set of the Pareto optimal solutions can be found as the nearest to, or better than μ_i solutions, by solving a minimax problem

$$\min_{x \in X} \max_{i=1,\ldots,k} \{\mu_i - \mu_i(f_i(\boldsymbol{x}))\}. \tag{3.12}$$

3.3.12. Directions of Further Research in Evolutionary Algorithms

Although the classical prototype of Evolutionary algorithms is based on a few fairly simple ideas, it serves as an extremely powerful and flexible platform for a great and ever-growing variety of implementations. The algorithm itself does not imply, nor

impose restrictions or limitations of any kind on the implementational details. Different representation schemes can range from the simplest flat binary encoding, to the most complex hierarchical and multi-dimensional schemes capable of rendering entire computer programs or neural networks. The user is free to choose or design any representation form that intuitively seems most appropriate and convenient for the problem at hand. The impressive toolbox of evolutionary operators contains solutions extending beyond the original proportional selection, single-point crossover, and random mutation. Fairly sophisticated operators are available that incorporate intelligent analysis into traditional analytical and heuristic methods. The expandable set of control parameters can be adapted, to guide the search toward the desired solution in more efficient way.

With all the advances in the development of the algorithms, it is fair to say, however, that the application of EAs to each particular problem is still, to a large extent, an art, rather than a routine procedure. A researcher, a scientist, or an engineer who wants to use the algorithms to solve a specific real-world problem, faces a challenge of choosing a slew of options and parameters that are not well defined, and can be very problem dependent. A lot of research has to be done before Evolutionary algorithms become an everyday, off-the-shelf problem-solving tool. The overview of research areas that are likely to play important role in further development of EAs identifies the following directions [25]:

1. Growing complexity of solution representation and morphogenesis. Much of the research in the EAs field has been done using relatively simple genotypic structures. Recent attempts to evolve more complex structures, such as neural

networks or computer programs show increasing demands for the development of more universal encodings, complemented by the process of morphogenesis. More complex and meaningful evolutionary operators have to be developed, which would be capable of producing structurally complex and non-trivial solutions.

2. Inclusion of operators not directly associated with the biological nature of evolution. One example of such operators is Lamarckian evolution allowing for characteristics acquired by an individual to be seamlessly incorporated into the learning experience of the population. Hybrid systems are appearing, where individuals go through a learning process, and then pass the acquired genotypic and phenotypic features on to the future generations.
3. Non-random mating and speciation have to play an increasingly important role, as the means of reducing the number of lethal or non-viable solutions produced during the mating process.
4. Works on decentralized and parallelized models have to draw increasing attention. Parallel implementation is capable of significantly reducing the high computational cost traditionally associated with evolutionary computations. Speciation, niching, punctuated equilibriums, and other multi-population techniques have to be rigorously investigated.
5. Adaptation and self-adaptation mechanisms controlling EAs parameters, and strongly affecting representation, recombination, and other features of the EAs process have to be studied and effectively implemented.
6. Various competitive and cooperative co-evolutionary models should bring increasing behavioral complexity and efficiency to evolutionary computations.

7. A big gap between the simplifying assumptions of theoretical EAs models and the actual complexity of their implementation has to be bridged. Strong theoretical results have to be produced, and powerful mathematical tools have to be designed, in order to describe the complex stochastic nonlinear systems that EAs appears to be.

3.4. Evolutionary Algorithms in Imaging Optimization

Evolutionary algorithms have been successfully applied to a variety of problems encountered in image processing, where they reportedly demonstrated superior performance over traditionally used methods. However, the total amount of research in this area is still relatively small, in comparison with other areas, like electrical and mechanical engineering; industrial and resource control; and many scientific applications. According to the comprehensive bibliography of the EAs research conducted in 1963-2004 [1], the number of publications in two areas, optics and image processing accounts for less than 8% of the total EAs publications. Moreover, analysis of 196 publications listed in [1], and related to imaging optimization, i.e., 2-D and 3-D image registration; object, target, pattern, and character recognition; scene interpretation and object classification; feature extraction; remote sensing; multi-sensor fusion; content-based image retrieval; and medical imaging, shows that the absolute majority of the research work (168, or 85.8%) utilizes the classical models of EAs, exploiting the advantages of the parallelism offered by these models. Genetic algorithms, Evolution strategies, Genetic programming, and Evolutionary programming, with random or heuristic-based initial population; roulette wheel, tournament, stochastic universal, or ranking selection; random

mutation; and random one-point, two-point, or uniform crossover are used in these works. Only 28 papers (14.2%) go somewhat beyond the classical EAs models. One half of them (14 papers, or 7.1%) reinforce evolutionary search with traditional techniques of local optimization, predominantly in the form of a simple hill climbing (13 papers), with only one paper using a more advanced Downhill Simplex Method, as an instrument of local search. Only 14 publications (7.1%) utilize other, more advanced techniques, like clustering and using sub-populations; multi-objective fitness evaluation; and more complex forms of solution representation or evolutionary operators. This section gives some typical examples of using EAs in various applications of imaging optimization, i.e., in image registration; pattern matching; scene interpretation and object recognition; feature selection; and human body registration.

3.4.1. Two-Dimensional Image Registration

The application of GAs to registration of medical x-ray images, in the presence of noise, is discussed in [34, 35, 55, 65, 102]. The proper alignment of images is critical for the success of the following digital subtraction angiography (DSA). In the process of registration for DSA, a mask image (i.e., a template) has to be transformed and aligned, to match the region of interest (ROI) in the contrast, or reference image, in order to eliminate artifacts introduced by the image motion. In [34, 55], the transformation between the images includes rotation, translation, and limited elastic motion. In [102], transformation is defined by the constrained x and y displacements of four corners of the ROI.

The emphasis is put on the quick approximate evaluation of increasingly larger number of candidate solutions during one iteration, as opposed to the highly accurate evaluation of a smaller number of chromosomes. In order to accomplish this task, the quality of the mapping between the images is evaluated on a sample set of points, rather than on the entire image. Statistical sampling techniques are used to randomly select the subsets of the image points involved in the fitness evaluation. The classical model of GAs is used, with single point crossover and small probability of random mutation. Experimental results obtained for the actual clinic images of arteries obscured with motion artifacts, validate the proposed GAs-based method of automatic registration of medical images, and show its high robustness and efficiency, in comparison with more traditional techniques used in medical imaging.

Close to image registration is the problem of model fitting discussed in [65], with application to medical images. The authors build a parametric model of the left ventricle (LV) of the heart, and attempt to find parameters of the model from ultrasound images of the heart, using GAs. Poor quality and incompleteness of clinical images; and presence of noise and artifacts are among the challenges encountered in the process of solving the problem, which makes the identification of the LV boundaries a difficult problem, even for medical experts. The LV model in [65] is defined by the set of six shape parameters, and by the set of four transformation parameters including image translations, rotation, and isotropic scaling in the 2-D polar coordinate system. All parameters are encoded as integer numbers in a chromosome. The fitness of the chromosome is evaluated as the strength of the observed boundary and its closeness to the shape predicted by the LV

model. The classical simple GA with remainder stochastic sampling as selection strategy (a more advanced variant of roulette wheel selection), single point crossover at the crossover rate $p_c = 0.6$, and random mutation at the rate $p_m = 0.005$ are used; the termination of the evolutionary search occurs at the maximum number of objective function evaluations, which is set to 5000. The performance of the algorithm is evaluated on a set of apical 4-chamber echocardiogram time sequences obtained from different sources. Authors report the high quality of automatic interpretation achieved with the model-based GA approach.

Hardware architecture of the parallel GA for image registration in computer vision systems is proposed in [160]. Images are transformed using translations in *xy* plane, rotation, and isotropic scaling. The fine-grain model of parallel GA with massive parallelism on a single VLSI chip is used [16], where the scope of interaction of each chromosome is limited to only its nearest neighbors. Fitness is evaluated as the number of matching pixels in the images. Tournament selection among the neighbors and two-point crossover are used as evolutionary operators. Local optimization with hill climbing technique on each parameter is added to the classical parallel GA model. Using hardware implementation, the authors achieved real-time registration of relatively small test images.

A two-phase GA-based image registration is proposed for 2-D grayscale satellite images in [17], particularly for the fast processing of massive data of the global Earth change. The transformation between the images includes translations in the *x* and *y* directions, and

rotation in the *xy* plane. The first phase is a sequential registration of low resolution compressed images; it allows one to obtain the approximate values of the transformation parameters. The images are compressed to 32×32-pixel format using wavelet transform. Despite the high compression factor of 256, the accuracy of the first phase is reportedly high. The second phase is a parallel refining of the results obtained from the first phase, applied to the actual, high resolution images. In order to reduce the computational cost of the registration, only a 128×128-pixel window in the 512×512-pixel images are considered for registration. The parallelism of the algorithm is achieved by the parallel evaluation of chromosomes on multiple processors using the master-slave model of the parallel EA [16]. The parameters of the transformation are encoded in a chromosome as integer numbers. The quality of the individual chromosome is evaluated as the correlation between two images *I* and *J*. Crossover is implemented in the form of the classical single point operator. The authors report that the algorithm is robust, accurate, and provides fast registration on a test suite of NASA and NOAA images, on a parallel Beowulf cluster of 50 Pentium Pro PCs, which are interconnected with a combination of GigabitEthernet and switched FastEthernet.

3.4.2. Three-Dimensional Image Registration

A 3-step registration process is used in [135], to solve the 3-D elastic multimodality registration of medical images, particularly of the brain images, obtained with the CT (computer tomography) scanner and the MRI (magnetic resonance imaging) technique. Two modified models of the classical Genetic algorithm are used for the first two steps.

The third step involves the classical local optimization applied to the best solutions obtained with the GA, in order to fine-tune the final result of the registration.

The paper assumes that image deformation is small, and rigid registration can serve as a good initial guess for a more elaborate second step of the process. During the first step, the modified version of the classical GA is applied, in order to find the rigid body transformation defined by six parameters: three translations and three rotations in the *x*, *y*, and *z* directions. A chromosome implements real encoding of the parameters. The algorithm uses fitness proportional selection enhanced with scaling and normalization, and a uniform crossover. Mutation is implemented as random bit flipping, with a small probability. The author of the paper modifies the classical model with a constraint based on the principle of Latin squares, to ensure that all subregions of the search space are fairly represented throughout the entire algorithm run, i.e., a certain minimum number of chromosomes for each subregion is present in the population. Following this constraint, mutation is restricted to small localities, thus acting as the operator of local improvement of the solution. During the second step of the method, the second modified version of the classical GA is applied, as a stochastic sampler aimed at reducing the search space at the final step of the procedure. The algorithm attempts to find the pairs of the corresponding points in the images, which uniquely define a non-rigid tri-linear transformation between the images. The list of potential candidates for correspondence is compiled from the best solutions obtained at the first step. A chromosome implements a binary encoding of the combination of eight points. The classical random mutation and single point crossover are used during the second step.

Application of Genetic algorithms to the 3-D free-form surface registration and model construction is considered in [19], with relation to medical imaging. Medical mages of the same object obtained with range sensors from different viewpoints have to be brought into a mutual correspondence. Investigators point out that the commonly used Iterative Closest Point (ICP) algorithm based on local optimization techniques requires a good initial guess, in order to find the correct transformation. Image transformation in [19] is defined by six parameters encoded in a chromosome: three rotation angles and three translations, along the x, y, and z axes. The quality of each chromosome is evaluated using fitness function defined as median Euclidean distance between the corresponding points of the images. The algorithm is a variant of a simple Genetic algorithm. The authors report several examples of the successful registration of large range images in a few minutes, on a regular PC with Pentium III processor running at 450 MHz. The GA-based registration does not require a good initial guess or any prior information about the correspondence (or feature) points, as opposed to the ICP algorithm.

3.4.3. Pattern Matching

An approach based on Genetic algorithms is proposed in [180], to solve an incomplete pattern matching problem for two point sets, $\boldsymbol{P}$ and $\boldsymbol{P}'$, under affine transformation T, when only a limited number of image points participate in the matching process. The entire parameter space is reduced by constructing "feature ellipses" of the point sets $\boldsymbol{P}$ and $\boldsymbol{P}'$. The feature ellipse of a point set $\boldsymbol{P}$ approximates the shape of $\boldsymbol{P}$, and is centered at the center of $\boldsymbol{P}$. Since three points define a valid affine transformation, a reference point triplet is selected on the feature ellipse of the point set $\boldsymbol{P}$. A chromosome x is

comprised of one of the possible binary coded point triplet of the point set $\boldsymbol{P}'$. The trial parameters of the transformation T are found between the reference point triplet in $\boldsymbol{P}$ and a chromosome in $\boldsymbol{P}'$. The chromosome x has to be found with GA, which minimizes the Hausdorff distance $H(T(\boldsymbol{P}), \boldsymbol{P}')$ between the point sets. A simple Genetic algorithm with roulette-wheel selection, single point crossover, and uniform mutation at a small rate is used to solve the matching problem. The initial population is generated at random, from the localities around the feature ellipse of the point set $\boldsymbol{P}'$. The final parameters of the transformation T are found by comparing the best found chromosome with the reference triplet. Theoretical and experimental results conducted on many point sets generated at random, show the efficiency of the proposed algorithm, in relation to the accuracy of the final transformation parameters found by the algorithm.

3.4.4. Semantic Scene Interpretation and Object Recognition

Given domain-specific knowledge, in a form of a high-level semantic classification of objects, semantic scene interpretation attempts to label the objects in the scene, according to the class to which they belong. Therefore, the solution to the problem represents a set of labels corresponding to different objects of the scene image. The automation of the interpretation task becomes a vital factor e.g., in processing of massive imagery data coming from satellites and digital communication systems.

A simple GA model is applied for recognition and classification of ocean currents, eddies, and continental shelf in segmented images of North Atlantic in [3]. A semantic net is introduced that establishes relationships between classes, in a form of fuzzy

predicates. The correspondence between the segments in the image of the scene and the semantic net of classes is binary-encoded in a chromosome. The fitness of the chromosome is a fuzzy function defined as the sum of all predicates that are satisfied in the given semantic net. The chromosome that has the maximum value of the sum of the predicates corresponds to the optimal classification scheme. A GA-based approach [3] to automatic classification and labeling of objects presented in the image is extended in [13, 123], where it puts into correspondence segments of the image obtained in the pre-processing stage, with the object classes of the semantic net representing domain-specific knowledge, and built prior to the interpretation task.

A parallel version of a simple GA with migration between subpopulations is used in [123]. A candidate solution is encoded in a chromosome, as a combination of indices, where each index identifies a potential class of the corresponding segment of the image. Therefore, each chromosome represents a potential mapping between the segments of the image and the object classes of the semantic net. The quality of the chromosome, i.e., its fitness is evaluated using feature comparator functions that compute the degree of correspondence between the feature sets of the image segments, and the feature sets of the object classes. The proposed method is applied to the classification of noisy and obscured natural scenes from oceanographic and meteorological data collections. Identification of ocean currents in infrared imagery of North Atlantic, and classification of clouds in AVHRR (Advanced Very High Resolution Radiometer) imagery of the Western U.S., show a high degree of accuracy.

3.4.5. Feature Selection

Many methods of pattern and object recognition or classification are based on the selection of a feature set characterizing some salient properties of an image or object of interest. The feature set has to be optimal, in a sense that it has to describe the most salient and important properties of image data, without the extraneous, redundant, or unimportant information that could increase the cost of recognition and classification, and could eventually result in creating overfitted models. Therefore, the problem of feature selection in imaging applications can be stated as an optimization problem [56, 93, 95, 103, 104, 141, 149, 159, 161, 177, 178].

Performance of a simple GA is compared in [149] against a few methods of feature subset selection traditionally used in statistical pattern classifiers: an exhaustive search, a generalized version of sequential search, and a variant of the branch and bound method. A chromosome is implemented as a binary string encoding a feature subset, with 1s corresponding to the presence of the respective feature, and 0s corresponding to the absence of the feature. This representation turns the problem into a combinatorial optimization in the space of all possible combinations of features. Crossover is a single point operator, and mutation is implemented as random flipping of one or more bits of a chromosome. Termination of the search occurs, after the pre-set number of tested combinations is exceeded. Tests conducted on simulated data and infrared imagery of real scenes show that the GA-based approach outperforms the other tested methods, in relation to both, the accuracy of the found feature subsets, and the computational

efficiency of the algorithm. The advantage of the GA-based method grows, as the dimensionality of the search space increases.

Application of GAs to selection of a character feature subset for an optical character recognition (OCR) system is investigated in [141, 159]. Each feature is assigned a cost associated with the average CPU time needed to compute this feature for a typical character. The algorithm searches for the feature subset that provides the optimal trade-off between the accuracy and the computational cost of the character recognition. The classical elitist model of GAs is utilized, with potential solutions being binary-encoded in a chromosome, in the form of different feature combinations. In addition to the traditional operations of crossover and mutation, a problem-specific rotation operator is introduced that right-rotates bits in a chromosome. The algorithm allows for a significant reduction of the computational cost of OCR, by producing the optimal subset, from the set of over 900 expert-defined features used to discriminate machine- and human-printed characters of Latin and Arabic character sets. The observed accuracy of the recognition does not degrade, and in some cases even increases, in comparison with the method that uses the entire feature set. The accuracy increase is attributed to the effect of removing some redundant or confusing features obscuring the recognition process.

Research has been done in [177, 178], on using Genetic algorithms to solve the problem of feature subset selection in medical applications. A chromosome encodes a feature subset represented as a binary string, where 1 corresponds to a particular feature presence, and 0 corresponds to its absence. The fitness of the chromosome is evaluated

by building an Artificial Neural Network, and evaluating its performance on a training set of patterns described by the selected feature subset. The performance of the ANN is evaluated based on two criteria, the accuracy and the cost of the classification. The accuracy of the classification is defined as the fraction of the correctly classified patterns in the test set. For a particular case of medical applications, the cost is defined as the cost of the measurements associated with the selected feature subset. Consequently, the fitness value of the candidate solution x is defined as

$$fitness(x) = accuracy(x) - (cost(x) / (accuracy(x) + 1)) + cost_{max}\,, \qquad (3.13)$$

where $cost_{max}$ is the sum of all costs associated with all features in the feature subset [177, 178].

A simple Genetic algorithm is utilized, with rank-based selection strategy, and the following values of the main control parameters:

- population size 50;
- number of generations 20;
- probability of crossover 0.6;
- probability of mutation 0.001;
- probability of selection of the highest ranked chromosome 0.6.

The experiments are conducted on large real-world datasets including those from machine learning databases and document classification systems. The results show that the GA-based approach outperforms most non-GA methods, in relation to both, the accuracy and cost of the selected feature subsets. The authors point out that GAs have

advantages, such as the capacity of simultaneous evaluating of multiple criteria (e.g., the accuracy and computational cost), and finding feature subsets that perform well for the particular inductive learning algorithms used to construct a pattern classifier based on decision tree or artificial neural network.

The problem of feature selection in unsupervised learning is formulated in [84], as a multi-objective optimization problem (MOOP), where each feature subset, together with corresponding feature clusters represent one solution approximating the Pareto optimal front. Four criteria (i.e., optimization objectives) are used to evaluate the quality of the solution:

- F_{within} favors dense clusters;
- F_{between} favors a good separation between the clusters;
- F_{clusters} favors creation of a smaller number of clusters;
- $F_{\text{complexity}}$ encourages minimization of the number of features.

To solve the formulated MOOP, a variant of Evolutionary algorithms called Evolutionary Local Selection Algorithm (ELSA) [105] is used that simulates a multi-agent system, in which agents interact with the environment under the constrained shared resources. Only mutation is used as a genetic operator modifying the candidate solutions. A mechanism of rewards and penalties is built into the algorithm, in order to favor and encourage the diversity of the solutions forming the Pareto optimal front. Investigators point out that ELSA provides a good coverage of the large space of potential feature combinations, together with the successful optimization involving evaluation of multiple criteria.

Moreover, good feature subsets are obtained during unsupervised learning, without the requirements of the preliminary model training.

3.4.6. Human Body Registration

The importance of recognition and registration of human body in real-world imagery is rooted in the variety of practical applications including such areas as automatic target recognition; security systems; robotic and computer vision; and industrial control. The complexity of the problem stems from the fact that human body is a flexible object that can arbitrarily move in the 3-D space, while changing its shape, i.e., its posture and gesture. From the registration point of view, one or more images (templates) Img_B of human body obtained from different camera views have to be mapped onto the same scene Img_R that plays a role of a common reference system. The problem of recognition and registration of human body with Evolutionary algorithms has been addressed by a number of researchers.

A variant of Genetic algorithms is used in [68] for the recognition of human posture in video sequences. A history of each individual pixel in a consecutive set of video frames is recorded, in a form of pixel value histogram. The silhouette of a moving body is recognized by analyzing noise that occurs in the histogram, caused by a temporary occlusion of the background pixels, when the moving body passes the corresponding location. Once the body silhouette is extracted, a hierarchical connecting tree (CT) is used to recognize a posture, in the following way. Each of 10 main parts of the body (e.g., head, trunk, etc.) is represented by a rectangle R, whose shape and spatial

orientation are defined by a 5-element array (x,y,w,h,α) of parameters, where x and y are the coordinates of R; w and h are the dimensions of R; and α is the angular orientation of R. Connecting relations between the rectangles, and the corresponding constraints imposed on their parameters are defined in the CT model of the body. The use of several simplified assumptions reduces the total number of parameters to 24. The search for the optimal values of the parameters providing the maximum degree of similarity between the CT model and the extracted body silhouette is conducted by means of the consecutive recognition of various parts of the body, followed by the elimination of the recognized parts from the search. The classical version of GAs is used for the search, with fitness function defined as the area difference between the CT model and the silhouette.

A skeleton model of human body is designed in [181], which consists of body segments connected by their respective joints. The spatial position of each joint is described by the parameter vector (α,β,γ) of rotation angles around the axes x, y, and z, respectively. Biomechanical constraints are imposed on the values of the rotation angles. A particular body posture is represented by a set of parameter vectors corresponding to the set of body joints. The difference between the postures of the skeleton model and the real body in the monocular image, is expressed in terms of energy function EF. The latter is comprised of orientation, length, and position deviations of the projected model from the real image features, as well as deviation of features in the i-frame, from the same features in the (i-1)-frame, denoted as the neighbor deviation. The motion reconstruction of the body from a series of monocular images is stated as a problem of finding the 3-D postures of the

skeleton model minimizing the energy function *EF*. The minimization problem is solved with the classical model of Genetic algorithms.

A model consisting of 10 parts connected by joints is used in [148], to track the frontal movement of human body. The body movement is defined by a parameter vector $(x,y,\theta_1,...,\theta_{10})$, where x and y are the coordinates of the body trunk, and $\theta_1,\ldots,\theta_{10}$ are the swiveling angles of the joints. A version of Evolutionary algorithms called Probability Evolutionary Algorithm (PEA) is used to minimize the difference between the model and the actual body movement. The algorithm is described in terms of probabilistic superposed bits of the encoded parameter vector, an observation stage, and an update stage. The observation stage is used to assign deterministic values to superposed bits, and the update stage is used to change observation probabilities of the bits, so the better fitness values can be observed more frequently.

3.5. Summary

Evolutionary algorithms have proven to be a fairly generic, flexible, and versatile framework for solving complex problems of global optimization and search encountered in real-world applications. In particular, the following advantages offered by the family of Evolutionary algorithms can potentially increase the computational efficiency of visual data processing required for solving complex imaging optimization problems:

1. Inherent explicit and implicit parallelism allowing for simultaneous evaluation of multiple candidate solutions to the problem can be effectively utilized for the real time processing on a multiprocessor hardware.

2. The algorithms' ability to work with nonlinear, non-continuous, non-monotonic, and non-differentiable objective functions is well suited for real-world image mapping applications, where object shapes might have irregularities and occlusions, and fitness function has to be computed numerically, for every trial transformation vector.
3. The principal evolutionary operators persistently mix different potential solutions drawn from the entire search space, which allows one to conduct the global search for the optimum transformation, within a sufficiently large parameter space.

Analysis of the current state-of-the-art of EAs application to imaging optimization, allows one to make the following conclusions:

1. The range of the EAs applications is still relatively small, in comparison with other methods of image processing. Evolutionary algorithms are still not considered among the mainstream imaging methods. This can be partially explained by the fact that using EAs is more of an art, than a routing procedure that can be used right out-of-the-box. Practitioners have to understand and master many intricate details, such as the flow of the algorithm, encoding schemes, evolutionary operators, control parameters, and different advanced EAs models.
2. Majority of imaging applications still rely on a simple classical variant of the algorithms, which is not efficient for many difficult real-world problems.
3. Significant advantages over conventional methods in solving problems for the rigid body and similarity transformations have been achieved via brute force approach, i.e., by exploiting the inherent parallel nature of Evolutionary

algorithms. Fewer significant results have been obtained for more general types of transformation, e.g., general affine transformation.

4. Advanced EAs implementations and hybrid models have to be analyzed and employed, in order to satisfy the time and complexity constraints imposed by the problems encountered in modern real-world imaging optimization. Furthermore, new directions leading to the increase of the computational performance of Evolutionary algorithms in the area of interest have to be proposed and investigated.

Chapter 4: Hybridization of Evolutionary Algorithms with Local Search, Mutation Control, and Selective Fitness Evaluation

4.1. Introduction

While being fairly general and versatile, the classical random model of EAs frequently exhibits low performance characteristics in solving many real-world applications including imaging optimization. In order to improve the computational performance of EAs, hybridization with other computational methods is needed, in line with the recommendations on further EAs research summarized in Chapter 3 (section 3.3.12). Section 4.2 of this Chapter investigates the performance of the classical model of Evolutionary algorithms on test imaging problems. The rest of the Chapter discusses the following main directions of hybridization that improve the EAs performance in imaging optimization:

1. Using a particular form of local optimization that alternates a well structured search algorithm with random search, in a cyclic manner, in order to refine the best solutions obtained with the global evolutionary procedure, and reduce the overall computational cost of local search (section 4.3).
2. Using bookkeeping techniques during mutation and local search, in order to keep track of the used subareas of the search space, and provide all subareas with a fair opportunity to participate in the evolutionary process (section 4.4).
3. Incorporating problem-specific knowledge into the algorithm, by autonomously extracting the most essential direct information about the problem at hand, in

order to reduce the computational cost associated with fitness evaluation (section 4.6).

Section 4.5 gives comparative analysis of the computational performance of the classical model of Evolutionary algorithms, and the model incorporating the aforementioned hybridization techniques. Section 4.7 concludes the Chapter with the summary of the findings.

4.2. Performance of the Classical Evolutionary Algorithm in Imaging Optimization

In order to test the performance of the classical model of Evolutionary algorithms commonly used in imaging optimization, and to identify the potential directions toward increasing the efficiency of the algorithms, computational experiments are conducted on registration of 2-D grayscale images. Two sets are tested using the classical evolutionary approach - see Figure 4.1. The top image in each set is a reference scene, and the bottom image is a section subject to registration. The first set (Figure 4.1, left) includes two 286×286-pixel infrared images of a truck taken from different camera views; the second image is then rotated counterclockwise. The second set (Figure 4.1, right) includes two 290×290-pixel images of an aircraft wing [70]. The larger image of the wing is obtained by cropping a rectangular section from the original image. The original problem of finding the correct alignment of two images in each test set is re-formulated as the optimization problem of finding the optimum 4-dimensional vector $\boldsymbol{V} = \{DX, DY, \theta, S\}$ that minimizes the difference F between the images. Here, DX and DY are the translations in the x and y directions, respectively, θ is the rotation angle in the xy plane, and S is the

isotropic scaling factor. The objective function F is defined as the squared difference of the gray values of the images, over the area of their overlap, in accordance with the statement of the imaging optimization problem given in Chapter 1 (section 1.2). The exact values of the translations DX and DY, and the value of the scaling factor S for the truck image are unknown; the rotation angle is known, and equals θ = 0.7854 (or 45°). The exact values of the mapping parameters for the wing image are known: DX = 170, DY = 240, θ = 1.5734 (90°), and S = 4.14.

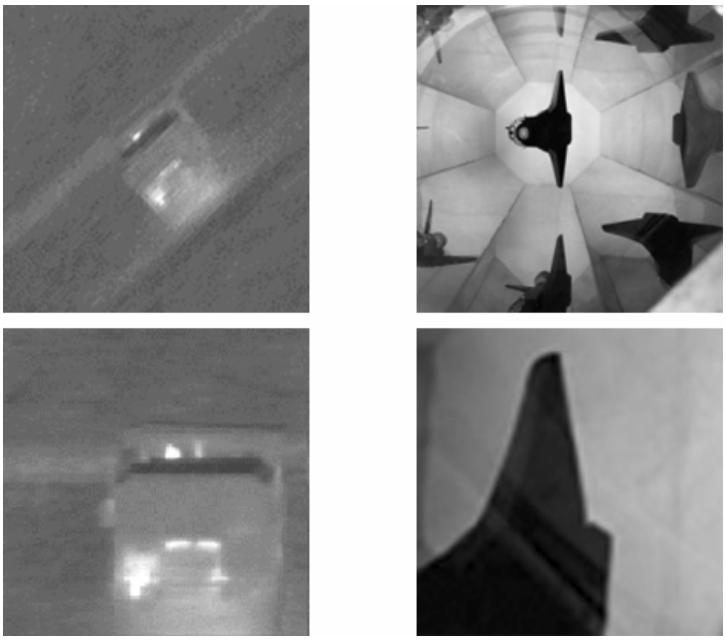

Figure 4.1: Two sets of test images: a truck (left) and a wing (right)

The classical Genetic algorithm with elitist reproduction, single-point crossover, random mutation, and roulette wheel selection is applied to both sets of images, as described in Chapter 1 (section 1.3.3). The initial chromosome population of the size P = 133 is selected at random from the entire search space, using the uniform probability

distribution. Elitist reproduction is implemented by copying 20% of the fitter chromosomes directly into the next generation. Mutants form 3% of the chromosome population; they are drawn from the entire search space at random, with the uniform probability distribution. The rest of the new generation is obtained by performing crossover on the current population, with the crossover probability $p_c = 1.0$.

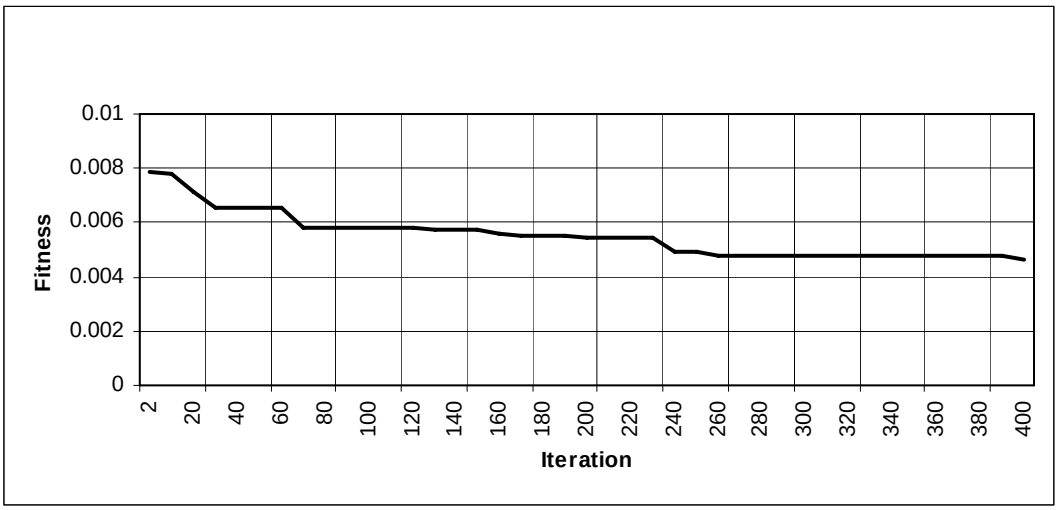

Figure 4.2: Performance of the classical elitist EA for the truck image

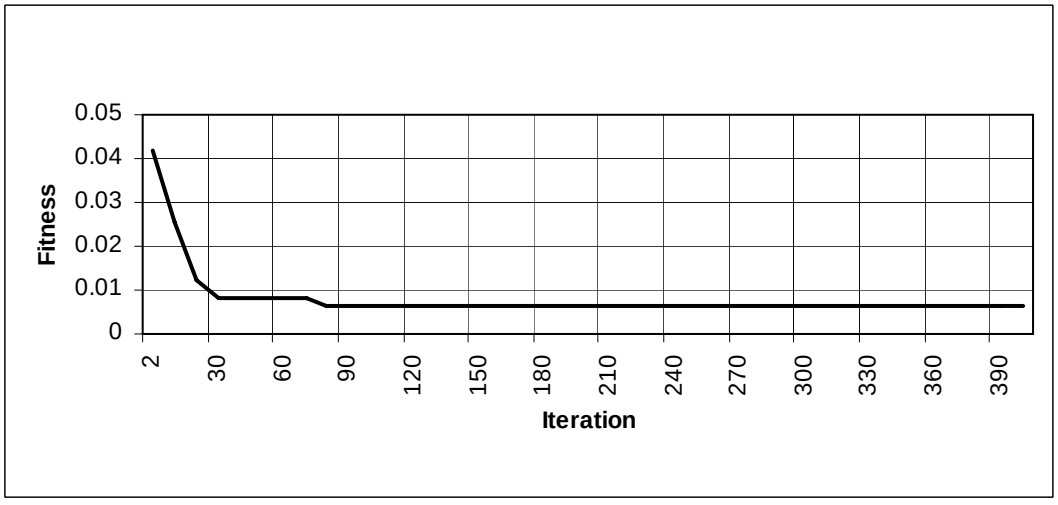

Figure 4.3: Performance of the classical elitist EA for the wing image

The charts in Figures 4.2 and 4.3 show the performance of the classical elitist strategy. The evolutionary process is terminated after 400 iterations, for the both image sets. The best fitness value for the truck image is $F = 0.00464$, with the parameter vector $\boldsymbol{V} = \{23, 158, 0.89664, 2.03323\}$. It is found at iteration 391, and remains the best solution until the process is terminated. The number of evaluations of the fitness function totals to 44213. The best fitness value for the wing image is $F = 0.00620$, with the parameter vector $\boldsymbol{V} = \{166, 245, 1.56509, 3.63680\}$. It is found at iteration 77, and remains the best solution, until the termination occurs. The total number of evaluations of the fitness functions is 8731. Both solutions are situated relatively close to their respective global minima; but they still require an additional amount of computational effort, in order to reach these minima.

Several observations can be made that explain a relatively slow convergence of the classical evolutionary strategy for the test problems. Firstly, reproduction and crossover, which are responsible for forming the majority (97% here) of the new population, do not introduce any new parameter values into the chromosome pool. It means that successful convergence of the algorithm mainly depends on how close parameter values of the initial random seed and additional mutants are to the global optimum values. If parameters are close enough to the optimum, then crossover would be able to produce a good combination of their values, and reproduction would carry them over to the next generation. In the most general case of the entirely random selection, the initial seed might not be successful. Then only mutation can improve the population. If the mutation rate is low, it might take a long time until “good” parameter values can been found.

Figure 4.4: Example of the propagation of fittest chromosomes throughout the entire pool (for the truck image)

Secondly, crossover tends to very fast propagate the fittest part of the chromosome pool across the entire population. The fittest parameter values are cloned and, in a matter of a few iterations, can spread across a significant fraction of the pool, even beyond its replicated elitist part. If propagating chromosomes point out to a local optimum solution, the iterative process can stall at this point, until some successful random mutant or result of the successful crossover appears in the pool, and brings a better solution. Figure 4.4 shows an example of the propagation of one of the intermediate fittest solutions, with the parameters $\boldsymbol{V}$ = {112, 35, 6.02412, 1.98746}, and fitness value F = 0.00658, for the truck image. The size of the reproduction pool is 30. The chromosome first appears as the fittest in the reproduction pool at the iteration 27; it generates a maximum number of clones 120 at the iteration 46. It has 99 clones at the iteration 64, when random mutation produces a better individual $\boldsymbol{V}$ = {40, 161, 0.91927, 2.49565}, with a lower fitness value F = 0.00582.

Finally, there is no mechanism built into the classical strategy that would allow one to control the granularity of the search. In the case of the wing image, the crossover operator produces a good solution $\boldsymbol{V}$ = {166, 245, 1.56509, 3.63680}, with the fitness value F = 0.00620, at the iteration 76; it stays at this point, until the procedure is terminated. The exact solution $\boldsymbol{V}$ = {170, 240, 1.5734, 4.14} could be easily reached, if there was a refining mechanism allowing for a more detailed analysis in the close vicinity of the best solution achieved by the global evolutionary search.

4.3. Local Optimization in Evolutionary Search

One of the efficient directions of improving the performance of EAs is Hybrid Evolutionary Algorithm (HEA) augmenting the classical model of EAs with traditional methods of local search and optimization [60, 80]. The idea behind HEA is relatively simple, albeit, very fruitful. Once the fitter chromosomes have been found at a current iteration, one can assume that they are located not so far away from one of the optimum solutions, whether local or global. Consequently, local neighborhoods surrounding the fittest chromosomes found by the global evolutionary search to date can be further explored in a greater detail, in an attempt to find possibly better individuals - see Figure 4.5. This is the basic idea behind the hybrid evolutionary approach. The idea of the local improvement of the solution finds its theoretical support, e.g., in the concept of Lamarckian evolution. The latter states that the individual's improvement achieved during the process of learning affects the individual's genetic structure, and can be mapped back to the genetic structure of the entire population [80].

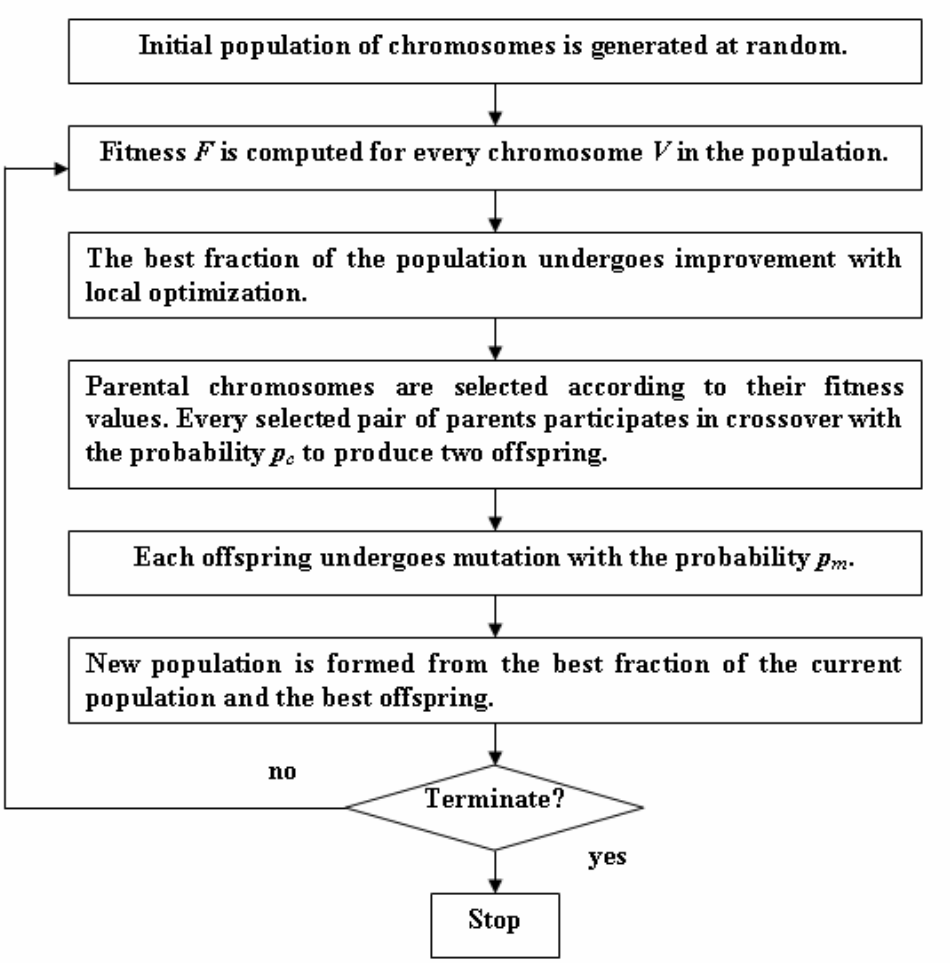

Figure 4.5: Combination of global and local search in Hybrid Evolutionary Algorithm

This section analyzes potential problems related to implementing local search in EAs, and discusses several methods of local search, with application to imaging optimization. First, the computational performance of gradient descent methods is investigated on a simple synthetic image set; these methods are commonly used in the EAs applications in imaging optimization. Then, more elaborate Conjugate Gradient Method and Downhill Simplex Method are tested, on a more difficult real imagery problem. A novel two-phase cyclic local procedure is proposed, which alternates random and Downhill Simplex Method local search procedures. Finally, different approaches to combining global and

local search strategies undertaken by Simulated annealing, Multi-start, and Evolutionary algorithms, are compared.

4.3.1. Problems Associated with Implementing Local Search in EAs

Potential candidates for a local search procedure in HEA have to be evaluated on the basis of two main criteria: robustness and computational performance. Robustness shows how readily the algorithm can adapt itself to various irregularities of the shape of the objective function. It also determines how close to the optimum solution the starting point of the search has to reside, in order for the algorithm to successfully descend (or ascend) to this optimum. Computational performance defines the quality of the final solution found by the algorithm, and computational cost, at which this quality has been achieved. Although improving the performance of EAs, the direct usage of local search gives rise to a few performance problems.

Firstly, local search adds a noticeable cost of the additional evaluations of the fitness function F attributed to this search. In a typical imaging application, the lion's share of the computational cost of EAs belongs to fitness evaluations. Every evaluation includes the transformation of the image, the pixel-wise comparison of the images, and the evaluation of the difference F between the images. The cost of the fitness evaluation grows, at least as $O(D^2)$, as the dimensions D of the images increase.

Secondly, the potentially good point that one decides to investigate in a greater detail, can be located in the vicinity of just one of the multiple local optima of the multimodal

objective function. In this case, all computational efforts spent on the local search (usually, hundreds of local iterations) would be wasted, if the area turned out to be away from the global optimum, and if the potentially good point was replaced with some other fitter point, with a different local neighborhood. The replacement might occur e.g., as the result of crossover or mutation in the classical EAs model, during one of the subsequent global evolutionary iterations.

Thirdly, even if the good solution is located not so far away from the optimum point, the actual distance might not be small enough, for the local search to take off and successfully find this optimum point. In the case of a multidimensional search, even a presence of a few satisfactory parameter values (e.g., translations DX, DY, and rotation θ) of the solution vector $\boldsymbol{V} = \{DX, DY, \theta, S\}$, cannot guarantee that the global evolutionary process will converge to the global optimum, if just one of the parameters (e.g., scaling factor S) is not satisfactory. The search most likely can stall for a certain amount of time at a point that is optimum only for the satisfactory parameters' subspace (i.e., sub-optimum point), as opposed to the entire search space. It might stay there, until some significant change of a non-satisfactory parameter value occurs (e.g., as a result of crossover or mutation), which will bring the fall-off parameter value close enough to the global optimum point having a better fitness than the point in a satisfactory parameters' subspace. Only then the process will be able to climb out of the local sub-optimum and continue to converge, hopefully to the global solution. Of course, it might equally happen that a significant change of the fall-off parameter value would throw the search process into another sub-optimum point.

There are two main categories of the traditional methods of local optimization [124] that can be potentially utilized in HEA:

- gradient-based methods engaging the derivatives of the objective function;
- direct search methods solely based on computing the values of the objective function.

Methods belonging to the first group are used in hybrid models, largely due to the fact that their properties and performance are well studied and understood by the mathematical community. Relatively simple Steepest descend algorithm, and more elaborate Conjugate Gradient Method [124], are among the popular choices in this category. Since these methods heavily rely on the accurate computation of the function derivatives, they pose certain, often severe demands on the shape and smoothness of the objective function and its derivatives. Optimization problems encountered in real-world imaging applications cannot guarantee that these demands can be satisfied: fitness function F describing the difference between images has inherently random nature, can be non-convex, and can exhibit shape irregularities. Both, the function and its derivatives, have to be computed numerically, the latter has to be computed using one of the known finite difference schemes.

Direct search methods have not received the same amount of attention, as their gradient-based counterparts, largely due to theoretical problems encountered by mathematicians who attempted to lay a coherent theoretical foundation for these methods. Only recently, direct search has seen a revival of interest, in light of the latest theoretical findings [87].

Among the optimization practitioners, however, these methods have always remained popular, because of their relatively low computational demands, high reliability, and often satisfactory final results. One of the popular choices of direct search is Downhill Simplex Method [116, 124, 176]. The DSM search does not require any additional function evaluations associated with computing partial derivatives of the function F. The method is often used in optimization practice, because of its robustness and, often, a fairly good convergence for non-differentiable and non-continuous objective functions; and a relatively small number of function evaluations per iteration, in comparison with other methods of direct or gradient search. Although the rigorous, good-for-all-cases theory proving the DSM convergence has not been developed yet, the method has attracted a good deal of attention, especially among practitioners. Since the method is tolerant to shape irregularities of the objective function, it makes it a good candidate for using in imaging optimization with HEA.

4.3.2. Performance of Deterministic Steepest Descent Method

The computational performance of one of the most popular local search procedures, a relatively simple deterministic steepest descent algorithm, is investigated on a sample imaging optimization problem. The algorithm is added to the classical elitist EA procedure, in the following way. At every iteration, all chromosomes in the replication pool are considered for local search. Every chromosome $\boldsymbol{V}_0$ is compared with its N neighbors $\boldsymbol{V}_i$ ($i = 1,\ldots,N$) drawn from the N-dimensional parameter sphere of a small radius r, centered at the chromosome $\boldsymbol{V}_0$. The gradient ∇F of the fitness function is used to measure the fitness gain for each neighbor:

$$\nabla F = (F_i - F_0) / d(\boldsymbol{V}_i, \boldsymbol{V}_0) , \quad (4.1)$$

where F_i and F_0 are the fitness values for the chromosomes $\boldsymbol{V}_i$ and $\boldsymbol{V}_0$; and $d(\boldsymbol{V}_i, \boldsymbol{V}_0)$ is the distance between the chromosomes. The neighbor that has the lowest negative value of ∇F replaces the original chromosome $\boldsymbol{V}_0$ in the replication pool.

Computational experiments are conducted on the same set of synthetic sensor readings that are used to test the comparative performance of the classical Evolutionary algorithm, Simulated annealing, and Tabu search, in Chapter 1 (section 1.3.4). The test set of two 256×256 – pixel synthetic images computed according to Formulae given in [12], is shown in Figure 1.2 of Chapter 1. The second image in the set is subject to the rigid body transformation defined by the a priori known vector of transformation parameters; the image has to be mapped onto the first image of the set. The minimization problem is stated in the 3-dimensional parameter space, with the parameter vector $\boldsymbol{V} = \{DX, DY, \theta\}$, and optimum solution $\boldsymbol{V}^* = \{91, 91, 2.78081\}$.

The same classical Genetic algorithm with elitist reproduction, single-point crossover, and random mutation is applied to the images, as used in section 4.2 of this Chapter. The population size is $P = 133$; the elitist reproduction is implemented by copying 20% of the fitter chromosomes; mutants form 3% of the chromosome population. The algorithm is augmented with the deterministic steepest descent search, where the nearest neighbors of every chromosome $\boldsymbol{V}_0$ in the replication pool change the values of the three parameters (i.e., the step size of DX, DY, and θ) by $\Delta = +/-1$. This corresponds to the 3-D parameter sphere with the radius $r = 1$. Altogether, six nearest neighbors have to be examined, and

their fitness has to be compared with the fitness of the original chromosome $\boldsymbol{V}_0$. The neighbor that has the lowest value of the fitness function (i.e., the fittest neighbor) replaces the original chromosome in pool. Although intuitively this approach seems to have to significantly improve the solution, the obtained final result $DX = 96$, $DY = 86$, $\theta = 2.83317$, with the minimum fitness value $F = 0.061$, and the weighted Euclidean distance from the exact solution $\delta = 0.114$, is slightly less accurate than the result without local search and correction, i.e., $DX = 89$, $DY = 84$, $\theta = 2.78081$, with the minimum fitness value $F = 0.062$, and the relative error $\delta = 0.086$.

4.3.3. Performance of Stochastic Steepest Descent Method

In the stochastic version of the steepest descent method, the strategy for selecting the fittest neighbor around the current chromosome $\boldsymbol{V}_0$ has been changed. The radius of the parameter sphere is increased to $r = 3$. Every chromosome in the replication pool is compared with its six neighbors that are drawn from within the sphere at random, with the uniform probability. The neighbor with the lowest negative value of the fitness gradient ∇F replaces the original chromosome in the pool.

The same classical Genetic algorithm with elitist reproduction, single-point crossover, and random mutation is applied to the images, as used in section 4.2 of this Chapter. The population size is $P = 133$; the elitist reproduction is implemented by copying 20% of the fitter chromosomes; mutants form 3% of the chromosome population. The algorithm is augmented with the stochastic steepest descent method. Figure 4.6 shows significant improvement of the fitness F convergence rate. A better solution $\boldsymbol{V} = \{DX = 94, DY = 88,$

$\theta = 2.80094\}$, with the minimum fitness value $F = 0.05674$, and the weighted Euclidean distance from the exact solution $\delta = 0.05$, is already found in iteration 12, as opposed to the value $F^* = 0.0620$ in 54 iterations of the classical EA. In iteration 36, the computed solution point $\boldsymbol{V} = \{DX = 91, DY = 93, \theta = 2.75192\}$ is very close to the exact global minimum; it has the minimum fitness value $F = 0.03796$, and the weighted Euclidean distance from the exact solution $\delta = 0.022$.

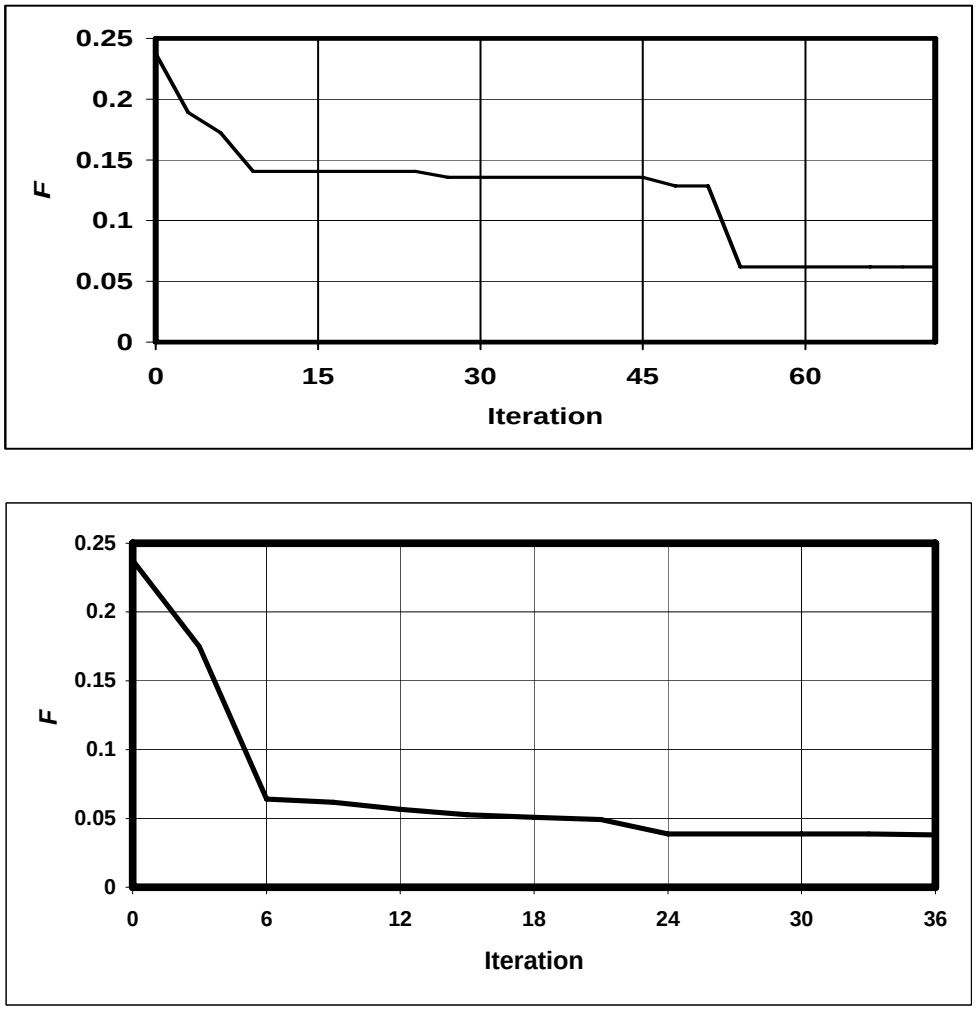

Figure 4.6: Performance of the classical (top) and hybrid (bottom) EA models

Interestingly enough, the increase of the radius r of the parameter sphere from $r = 3$ to $r = 6$, does not improve the convergence rate of the algorithm. On the contrary, it takes slightly longer to reach similar to $r = 3$ results: the search is able to find a satisfactory point $\boldsymbol{V} = \{DX = 91, DY = 86, \theta = 2.78386\}$ in iteration 18, with the minimum fitness value $F = 0.05809$, and the relative error $\delta = 0.056$. The closest to the global minimum solution $\boldsymbol{V} = \{DX = 91, DY = 93, \theta = 2.75242\}$ is found in iteration 48, with the minimum fitness value $F = 0.0387$, and the relative error $\delta = 0.022$.

Experimental results for different values of the radius r of the parameter sphere show that the optimum value of the radius r exists that can improve the convergence rate with the local gradient search. A smaller than the optimum value of r does not significantly affect the current values of the parameters. The value of r larger than the optimum, can result in the degradation of the quality of the final solution, and might interfere with the evolutionary character of the iterative process.

4.3.4. Performance of Conjugate Gradient and Downhill Simplex Methods

In this section, two advanced techniques are evaluated, as potential candidates for local search and correction of the reproduction pool in HEA. These techniques are Conjugate Gradient Method (CGM), and Downhill Simplex Method (DSM) [124]. The first technique, CGM, utilizes the first derivatives of fitness function. The partial derivative of the fitness function F along the component V_k of the parameter vector $\boldsymbol{V}$ is computed here using the first-order forward finite difference scheme:

$$\partial F / \partial V_k = (F(V_k + \Delta V_k) - F(V_k)) / \Delta V_k \ , \qquad (4.2)$$

where ΔV_k is the minimum step along the component V_k [124]. This technique requires one additional computation of fitness for every component V_k of the parameter vector $\boldsymbol{V}$. The second technique, DSM, does not require computing the function derivatives.

Computational experiments are conducted on test sets of real 2-D grayscale images. Each test set includes images of an object and a scene. The object represents a part of the scene that undergoes partial affine transformation *A*, and has to be mapped back onto the scene. For the comparative evaluation, the same test sets of images of a truck and aircraft wing are used that are used for the evaluation of the classical Evolutionary algorithm at the beginning of this Chapter – see Figure 4.1. A more difficult registration problem is considered, where the sought 4-dimensional vector $\boldsymbol{V}$ of transformation parameters includes the translations *DX* and *DY*, the rotation angle θ, and the scaling factor *S*. Both search techniques, CGM and DSM, are evaluated on the task of refining the global solution, when the starting point of the local search is placed at a specified distance from the global minimum point. In the case of CGM, the starting points are shifted from the exact location (i.e., from the global minimum point) by 1% through 9% of the image size. Since DSM required five vertices for the initial simplex, in this case of the dimensionality of the search space $N = 4$, two schemes of vertex placement are used. In the first scheme, the points are evenly shifted from the global minimum by 7%, 8%, 9%, 10% and 11% of the image size. In the second scheme, the points $\boldsymbol{V_i}$ are placed around the minimum $\boldsymbol{V^*}$ with the local step $\lambda = 10\%$ of the image size, according to Formula

$$\boldsymbol{V_i} = \boldsymbol{V^*} + \lambda\, \boldsymbol{e_i}\,, \tag{4.3}$$

where λ is the initial step along the unit vector $\boldsymbol{e_i}$, in the *i*-direction.

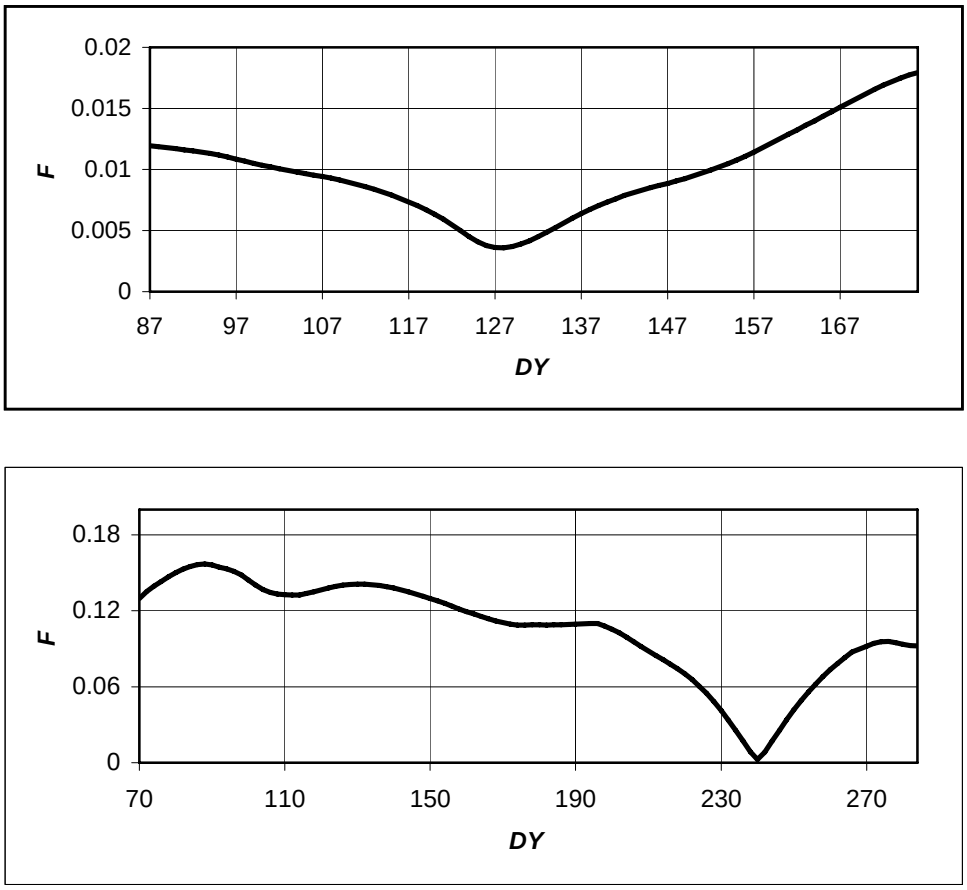

Figure 4.7: Fitness function *F* along the *y* axis near the optimum solution, for the truck (top) and the wing (bottom) images

Figure 4.7 shows sample graphs of the fitness function *F* along the *y* axis (with other three components fixed at their optimum values), in the vicinity of the optimum solution, for both objects. In the case of the truck, the function is convex and sufficiently smooth, which makes it well suited for evaluating the gradients. The shape of *F* becomes closer to parabolic, as the value of the translation *DY* approaches the optimum point. Based on these geometric properties of the function, one can expect that the derivative-based CGM can exhibit good performance for the truck. On the contrary, the fitness function of the wing is significantly non-convex, and almost exhibits a singularity at the optimum point.

Only if the point is very near the optimum solution, the shape of *F* becomes somewhat close to parabolic. Consequently, one can expect that CGM can perform well only if the starting point of the search is very close to the optimum.

	Truck			**Wing**		
	1%	**3%**	**9%**	**1%**	**3%**	**9%**
Fitness (×10 ^{5})	398	412	812	599	1604	fail
Iterations	58	101	119	60	40	fail
Function evaluations	91	132	168	78	49	fail
Derivative evaluations	61	106	125	63	42	fail
Total evaluations	152	238	333	141	91	fail

Table 4.1: CGM performance at the initial distance of 1%, 3%, and 9% from the optimum solution

	Truck		**Wing**	
	7% - 11%	**10%**	**7% - 11%**	**10%**
Fitness (×10 ^{5})	391	338	253	308
Iterations	40	137	42	127
Function evaluations	46	148	50	139
Derivative evaluations	-	-	-	-
Total evaluations	46	148	50	139

Table 4.2: DSM performance at the initial distance range of 7% - 11%, and at the radial distance of 10% from the optimum solution

The results of the computational experiments presented in Tables 4.1 and 4.2 support the preliminary arguments. As expected, CGM shows reasonable stability for the truck object, within the tested range of 1% - 9% of the placement of the starting point. The quality of the final solution degrades, as the distance increases: the computed minimum fitness value grows from $F = 0.00398$ at the 1% distance, to $F = 0.00812$ at the 9% distance. In the case of the wing, the performance of CGM quickly deteriorates, as the starting point moves away from the optimum: the computed minimum value of the fitness function increases from $F = 0.00599$ at the 1% distance, to $F = 0.01604$ at the 3% distance. The algorithm breaks down and quickly diverges beyond the 3% distance range.

Downhill Simplex Method is much less sensitive to the shape of the fitness function, whether convex or non-convex; it displays noticeable robustness in both, the truck and wing cases. The algorithm provides good final results, when the initial simplex vertices evenly placed at the 7% - 11% distance, and when the vertices are initially placed around the optimum value at the 10% distance. The performance characteristics of DSM are significantly superior to that of CGM. The computed minimum values for the truck, $F = 0.00391$ and $F = 0.00338$, at the 7% - 11%, and at the 10% distance ranges, respectively, are even better than the value $F = 0.00398$, at the 1% distance obtained with CGM. For the wing object, DSM finds very good optimum values $F = 0.00253$ and $F = 0.00308$ during both runs, whereas CGM fails to obtain any result beyond the 3% initial distance from the optimum. Moreover, the number of function evaluations with DSM in all considered cases is significantly smaller than the total number of function evaluations obtained with CGM, e.g., 46 at the range 7% - 11%, against 333 at the distance 9%, for

the truck object. The correct mapping between the test images found with DSM is shown in Figure 4.8.

The results of this group of experiments are clearly in favor of using direct search methods, particularly Downhill Simplex Method, over gradient-based methods like CGM, as the former can effectively deal with nonlinear, non-convex, numerically computed fitness functions that routinely arise in real-world imaging applications.

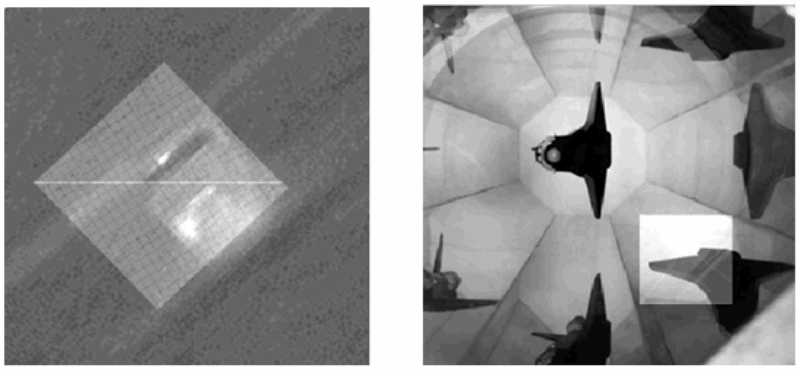

Figure 4.8: Successful mapping of the template objects onto their respective original scenes: the truck (left) and the wing (right) images

4.3.5. Two-Phase Cyclic Local Search Algorithm

As it is mentioned in section 4.3.1 of this Chapter, it is not very efficient to use local search in a direct manner, due to the fact that the current satisfactory solution under analysis can be only a temporarily fit solution, and might be replaced by a fitter chromosome during the subsequent iterations. Instead, a somewhat conservative and timesaving approach to the local correction of chromosomes in the reproduction pool is

proposed in this section. Local search is implemented as two alternating phases - DSM and random search, – as shown in Figure 4.9.

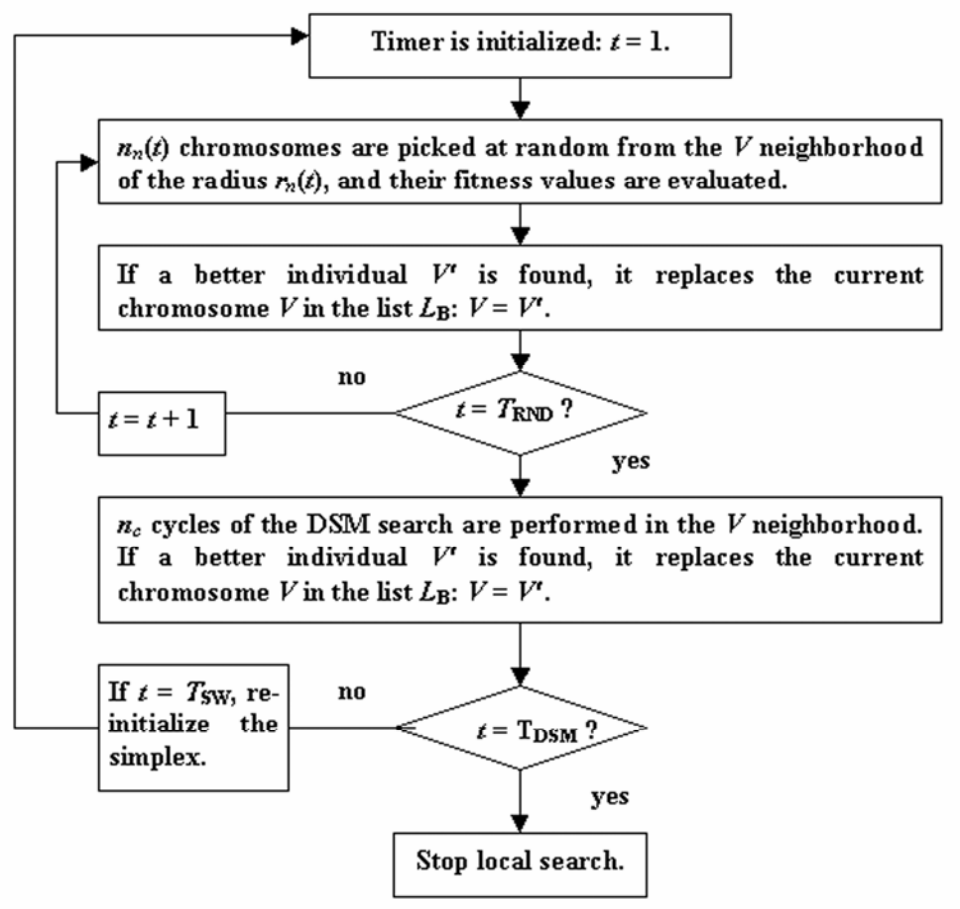

Figure 4.9: Two-phase cyclic random/DSM local search

In the first phase of the algorithm, a random search for a better individual is performed within the local neighborhood of every chromosome $\boldsymbol{V}_k$ in the reproduction pool (i.e., around one of the fitter chromosomes). Figure 4.10 illustrates the mechanism of the random local search on a sample 2-D model, where n neighbors of the targeted chromosome $\boldsymbol{V}_k$ are drawn at random from its expanding neighborhood. The

neighborhood is modeled with an N-dimensional hyperellipse H built around $\boldsymbol{V}_k$ as a center.

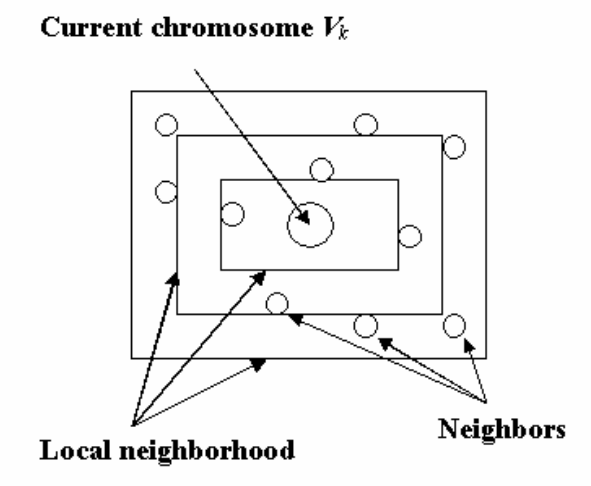

Figure 4.10: Sample 2-D model of random search within the expanding neighborhood of the targeted chromosome $\boldsymbol{V}_k$

As the global evolutionary search progresses, the chromosome $\boldsymbol{V}_k$ might retain its position in the reproduction pool. This can be an indicator that $\boldsymbol{V}_k$ is a local or, possibly, the global minimum solution. The safest and the most conservative assumption is that $\boldsymbol{V}_k$ is only a local minimum, and the process might need to climb out of this point. In order to increase the probability of the successful climbing, the radius of the hyperellipse H is set to grow, in accordance with the duration of time t that the point $\boldsymbol{V}_k$ retains its position in the pool. In this work, the radius ρ of the neighborhood is a linear function of time t:

$$\rho = \rho(t) = \xi(t - t_0), \tag{4.4}$$

where ξ is the initial interval of the radius increase (i.e., the local search step) at the beginning of the search, for $t = t_0$. The number η of the trial points drawn from the

neighborhood is also set to grow linearly with the time t, in order to provide the sufficient trial pool of neighbors:

$$\eta = \eta(t) = \zeta (t - t_0), \tag{4.5}$$

where ζ is the initial number of neighbors at $t = t_0$.

The actual location of the trial neighbor is found using the uniform probability distribution inside the hypershell S, with the thickness $\Delta\rho$ defined as follows:

$$\Delta\rho = \rho (t_i) - \rho (t_{i-1}), \tag{4.6}$$

where $\rho (t_{i-1})$ and $\rho (t_i)$ are the radii of the expanding hyperellipse H at the successive time intervals t_i and t_{i-1}.

If a neighbor $\boldsymbol{V'}$ is found that has better fitness value than $\boldsymbol{V}_k$ (i.e., $F' < F_k$), then $\boldsymbol{V'}$ replaces the current chromosome $\boldsymbol{V}_k$ in the reproduction pool. If no neighbor has been found that has a better value of F than the chromosome $\boldsymbol{V}_k$, the latter retains its position in the reproduction pool. The random search in the local neighborhood of the chromosome $\boldsymbol{V}_k$ continues, until the time t reaches the first time threshold $t = T_{\mathrm{RND}}$. Since the initial values of the factors ξ and ζ in Formulae (4.4) and (4.5), as well as the threshold value T_{RND} are pre-set, a full control can be maintained over the number of fitness evaluations, allowing to control the amount of computational resources spent during this phase of the local search.

If the chromosome $\boldsymbol{V}_k$ still remains in its position in the pool after the threshold T_{RND} has passed, there is a fairly good chance that the chromosome can be in the vicinity of a local,

or the global minimum solution. Therefore, in the second phase of the algorithm, the search inside the prospective neighborhood area needs to be refined using a more efficient technique, Downhill Simplex Method. The algorithm is an iterative procedure maintaining a non-degenerate (N +1)-dimensional simplex, at every iteration in the N-dimensional search space [124]. The vertices $\{\boldsymbol{V}_1, \boldsymbol{V}_2, \ldots, \boldsymbol{V}_{N+1}\}$ of the simplex, together with their respective function values $\{F_1, F_2, \ldots, F_{N+1}\}$, form a set of approximations to the N-dimensional parameter vector $\boldsymbol{V}$ and objective function F of the minimization problem. The algorithm attempts to change the worst vertex of the simplex to a better position at every iteration, in such a way that the function value would decrease. As the procedure iteratively progresses, the simplex is continuously moving and shrinking. The process is terminated when the size of the simplex or the difference between the function values at the simplex vertices becomes very small.

The algorithm uses four coefficients $\{\alpha, \beta, \gamma, \delta,$ to change the location of the simplex vertices. The standard values for the DSM coefficients are (-1, 2, 0.5, 0.5). When DSM is applied to the chromosomes in the reproduction pool in the second phase of the local search algorithm, its work is suspended after a specified number n_c of complete cycles of the DSM procedure have finished. The complete cycle means that the initial simplex has shrunk, and all (N+1) vertices have changed their locations gaining the decrease of their respective fitness values. The vertex having the best fitness value replaces the current chromosome $\boldsymbol{V}_k$ in the reproduction pool. The local DSM refinement is repeated for every chromosome in the reproduction pool, at the current global iteration.

After a pre-set switch threshold $t = T_{SW}$ has passed, the local procedure switches from DSM back to the random search, which might re-locate the vertices of the simplex obtained by DSM, before the next DSM cycle will resume its work. The alternation of the two search techniques, random and DSM, allows to periodically re-evaluate the position of the simplex in the local search space every time the simplex has shrunk, thus decreasing the chance of its converging to a sub-optimum solution. If the chromosome $\boldsymbol{V}_k$ retains its position in the reproduction pool after a second pre-set time threshold $t = T_{DSM}$ has passed, the chance is very high that the individual can be in the closest vicinity of the global minimum solution. At that moment, the DSM algorithm is allowed to complete its work, without further interruption.

The algorithm maintains a variable-length global list L_B of the fitter, i.e., having lower fitness values F, chromosomes, such that the local neighborhood of each chromosome $\boldsymbol{V}_k$ in the list is explored in a greater detail, in an attempt to find a better individual. The search procedure keeps track of the neighbors selected during the random search, by building a binary tree with the targeted chromosome $\boldsymbol{V}_k$ as the initial root. Once the specified number of neighbors has been evaluated, the tree is rotated to the right; an individual $\boldsymbol{V}_j$ with the lowest value of the fitness function F_j replaces the current chromosome $\boldsymbol{V}_k$, and becomes a new root of the tree. When the procedure switches back to DSM, it chooses the chromosome $\boldsymbol{V}_j$ as the first vertex of the initial simplex. The other N vertices of the simplex are picked from the right brunch of the tree; these vertices have the next N lowest values of the fitness function. This selection technique ensures that the

best-known-to-date simplex in the local neighborhood is formed, as the starting point for the resumed and re-positioned DSM search.

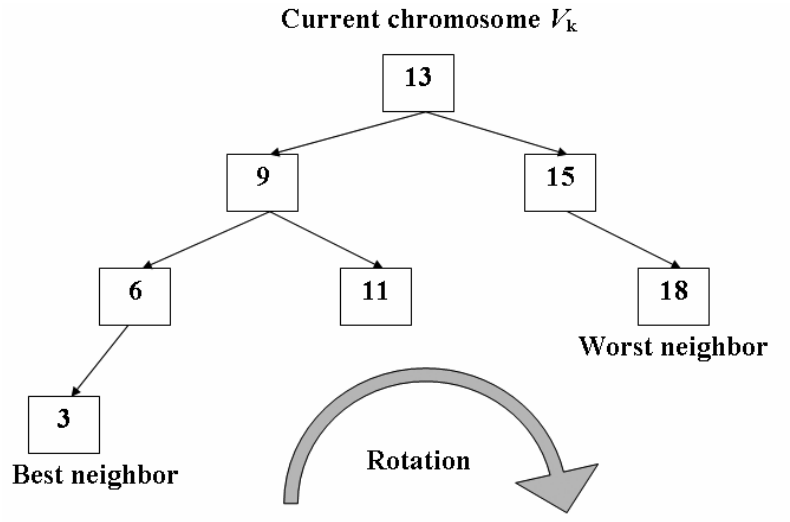

Figure 4.11: Sample binary tree of the neighborhood of the current chromosome $\boldsymbol{V}_k$

Figure 4.11 illustrates the mechanism that keeps track of the chromosomes located in the neighborhood of the current chromosome $\boldsymbol{V}_k$. In Figure 4.11, the square boxes represent the visited chromosomes in the neighborhood; they are labeled in accordance with their respective fitness values. The left-most leaf of the tree corresponds to the chromosome with the best value of F, which will replace the targeted chromosome $\boldsymbol{V}_k$ during the next cycle of the DSM search.

In summary, the proposed algorithm of a two-phase cyclic local search has the following distinctive features:

- using direct (here, DSM), as opposed to gradient-based methods of local search;

- adding randomness to the rigorously defined algorithmic framework of the chosen DSM procedure;
- engaging repetitive and partial (i.e., cyclic), rather than complete local search;
- building a tree structure, in order to keep track of the visited points in the local search space.

The algorithm alternates random and DSM search procedures, in order to help defy the performance problems related to the common usage of local search in HEA which are discussed in section 4.3.1 of this Chapter. The cyclic implementation of the DSM search reduces the potential waste of the computational resources spent by DSM, in the case when the targeted chromosome $\boldsymbol{V}_k$ will be replaced by the global search with a fitter candidate located outside the current local neighborhood. Occasionally, the classical DSM search can converge to a sub-optimum solution; the addition of the random search helps re-position the simplex at intermediate steps in the most advantageous manner, and thus prevent it from falling into a sub-optimum solution. Randomness, however, introduces a disturbance into the otherwise structured and algorithmically unambiguous simplex movement. Some of the points of the local search space that have already been visited by the simplex can be re-visited, as the result of the added randomness. Moreover, the simplex movement can even slip into a repetitive loop over the same sub-region. Monitoring the local search space and keeping track of the locations already visited by the simplex becomes an important issue, and the proposed mechanism based on the binary-tree model of the local neighborhood helps resolve this issue.

4.3.6. Computational Experiments with Two-Phase Local Search

In this section, a modified local search with the proposed two-phase (random/DSM) procedure is compared with the regular DSM procedure described in [124]. The same test set of a wing shown in Figure 4.1 is used, with the initial placement of the simplex vertices evenly, within the 7% - 11% range from the optimum solution. The results presented in Table 4.3 show a 32% reduction (from 50 to 34) of the total number of fitness evaluations, when the modified algorithm is used. The final optimum value F = 0.00216 found with the modified version is even slightly better than the final value F = 0.00253 found with the regular DSM.

	Regular DSM	**Modification**
Fitness ($\times 10^{5}$)	253	216
Cycles	8	4
Evaluations	50	34

Table 4.3: Performance characteristics of the regular DSM and its two-phase modification

The results indicate that the proposed two-phase modification of the local search procedure including random re-evaluation of simplex vertices, alongside with maintaining the record of the visited points in the simplex neighborhood, can speed up the overall convergence of local search, while retaining the high quality of the computed optimum values. The more robust convergence of the random/DSM search, together with

a relatively small number of function evaluations makes it a good choice for the local refinement in the hybrid evolutionary strategy.

4.3.7. Conclusions on Local Search in Evolutionary Algorithms

Although the hybrid EA model can significantly improve the overall performance of evolutionary search, the direct usage of methods of local optimization as simple plug-ins gives rise to a few performance problems. One of the problems is related to the noticeable cost of the additional evaluations of fitness function attributed to the local search. Another problem stems from the fact that in real-world imagery, fitness function commonly has multiple local minima, in which case the computational resources spent on exploring the local neighborhood of a particular chromosome will be wasted if the chromosome happened to belong to one of such local minima, and was discarded at a later stage of the evolution. The third problem is associated with the actual distance from the investigated chromosome to the global optimum solution: the distance might not be sufficiently small for the local search to successfully take off and descend to the desired optimum point.

Computational performance of local search can be potentially improved by applying the following techniques incorporated in the proposed two-phase cyclic local search algorithm:

- using direct vs. gradient-based method, in order to accommodate shape irregularities of fitness function arising in real-world imaging applications; in

particular, Downhill Simplex Method is recommended, because of its relatively high performance;

- adding randomness to the chosen local DSM procedure, thus periodically re-positioning the search, in order to prevent its convergence to a sub-optimum solution;
- creating a tree-like structure of the local neighborhoods and engaging memory bookkeeping operations, in order to keep track of the used local search space, and to prevent its recycling;
- using cyclic vs. complete local search, in order to cut down on excessive computational cost of local operations when a targeted chromosome is discarded at a later stage of the global evolutionary search.

4.3.8. Comparative Analysis of Strategies Combining Global and Local Search

This section compares the computational performance of three stochastic methods of global optimization [30] that differ in the strategy they use to combine global and local search: Simulated annealing, Multi-start, and Hybrid Evolutionary Algorithm – see Table 4.4.

Simulated annealing starts with a random initial candidate solution in the search space, and uses random hill-descending local search. In order to avoid trapping in local minima, the algorithm occasionally climbs up the local hills, with the probability defined by Boltzmann distribution, and continues the global search. It is the occasional deviation from the hill-descending search that expresses the essence of the global search strategy of

Simulated annealing. In order to make a fair comparison of all three methods, a simplified parallel version of Simulated annealing is utilized, where multiple instances of the algorithm are concurrently started using random seed of initial points in the entire search space.

Strategy / **Method**	**Global search**	**Local search**
Simulated annealing	Occasional uphill climbing driven by the Boltzmann distribution	Random hill-descending
Multi-start	Random seed of the starting points; keeping track of the visited points	One of the traditional local search techniques
Hybrid EA	Search driven by evolutionary operators	One of the traditional local search techniques

Table 4.4: Three strategies of combining global and local search

Multi-start algorithm starts with an initial random set of points in the search space, usually with the uniform probability distribution. The method then launches independent local search in the neighborhood of each initial solution using one of the known local search strategies. Multi-start algorithm has a global search strategy similar to Tabu search, i.e., it keeps track of the visited seeded points in the search space. When the

number of starting points approaches infinity, the best solution obtained by the local search procedures is the global optimum. The success of the algorithm depends on the number and locations of the starting points; and on the chosen strategy of the local search. The sufficient number and favorable distribution of the starting points are critical for the overall ability of the algorithm to reach the global solution. The local search strategy significantly affects the speed of the algorithm. In these comparative experiments, Downhill Simplex Method is utilized as the local search engine in Multi-start algorithm.

Hybrid Evolutionary algorithm combines global search based on the evolutionary strategy and evolutionary operators, and local search based on the traditional techniques of local optimization. Much like Multi-start, the algorithm starts with a set of initial candidate solutions in the entire search space. Unlike Multi-start, the algorithm proceeds using the global strategy of mixing and recombining different solutions, re-evaluating and comparing current and new solutions; and promoting the best candidates. Within the global search, the algorithm attempts to improve the better solutions by conducting local search within their close neighborhoods. The two-phase cyclic local search procedure is utilized in these experiments, which is proposed in section 4.3.5 of this Chapter.

A set of real 2-D grayscale images [70] is tested, in order to compare the performance of the three stochastic methods discussed in this section – see Figure 4.12. The set includes the 256×256-pixel scene containing several objects (Figure 4.12, left), and the 170×128-pixel object image (Figure 4.12, right), cropped from the scene and subject to the partial

affine transformation. The object image has to be registered back onto the scene. The optimum values of the components of the 5-dimensional transformation vector $\boldsymbol{V} = \{DX, DY, \theta, SX, SY\}$ are $\boldsymbol{V}^* = \{150, 212, 1.57, 4.14, 2.07\}$, i.e., the object is distorted by stretching along the x axis, in addition to the similarity transformation. In order to make the fair comparison across all three algorithms, they all are started with the same initial distribution of the candidate solutions: 133 starting points are selected at random, with the uniform probability distribution. The performance of the algorithms is evaluated on the basis of the following three criteria:

- factual probability of obtaining the global optimum solution (the success rate);
- number of function evaluations required to reach the final solution;
- quality of the best final solution obtained by the algorithm (the best fitness function value F).

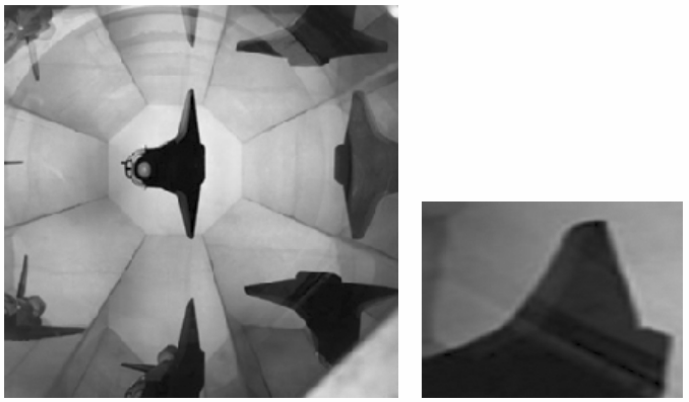

Figure 4.12: Original scene (left) and the object template (right)

For the parallel Simulated annealing, two cooling schedules are tested:

- slow schedule, with the rate of temperature decrease $R_T = 0.95$, maximum number of position modifications $M_{max} = 800$, maximum number of inner iterations $I_{max} = 4000$, and maximum number of outer iterations $O_{max} = 90$;
- fast schedule, with the rate of temperature decrease $R_T = 0.9$, maximum number of position modifications $M_{max} = 500$, maximum number of inner iterations $I_{max} = 2500$, and maximum number of outer iterations $O_{max} = 60$.

In both schedules, the value of the initial temperature is set to $T_0 = 1.0$, and Boltzmann's constant takes its value from the set (0.005, 0.01, 0.05, 0.1, 0.5, 1.0). Charts in Figure 4.13 show the success rate, the average number of fitness evaluations over all successful solutions, and the average best fitness over all successful solutions, for different values of Boltzmann's constant. As one can see, the value of Boltzmann's constant has significant impact on the success rate and number of function evaluations; it has much smaller impact on the best achieved minimum value of the objective function F. The success rate for the slow cooling schedule grows faster than the success rate for the fast cooling schedule, up to the break point in the chart corresponding to the value of Boltzmann's constant $k = 0.1$. At this point, the success rate for the slow schedule equals 0.429, and the success rate for the fast schedule equals 0.271. After the break point, the success rate for the slow schedule decreases, while the success rate for the fast schedule shows significant increase; the respective values of the success rate at the point corresponding to $k = 1.0$, equal 0.271 and 0.451.

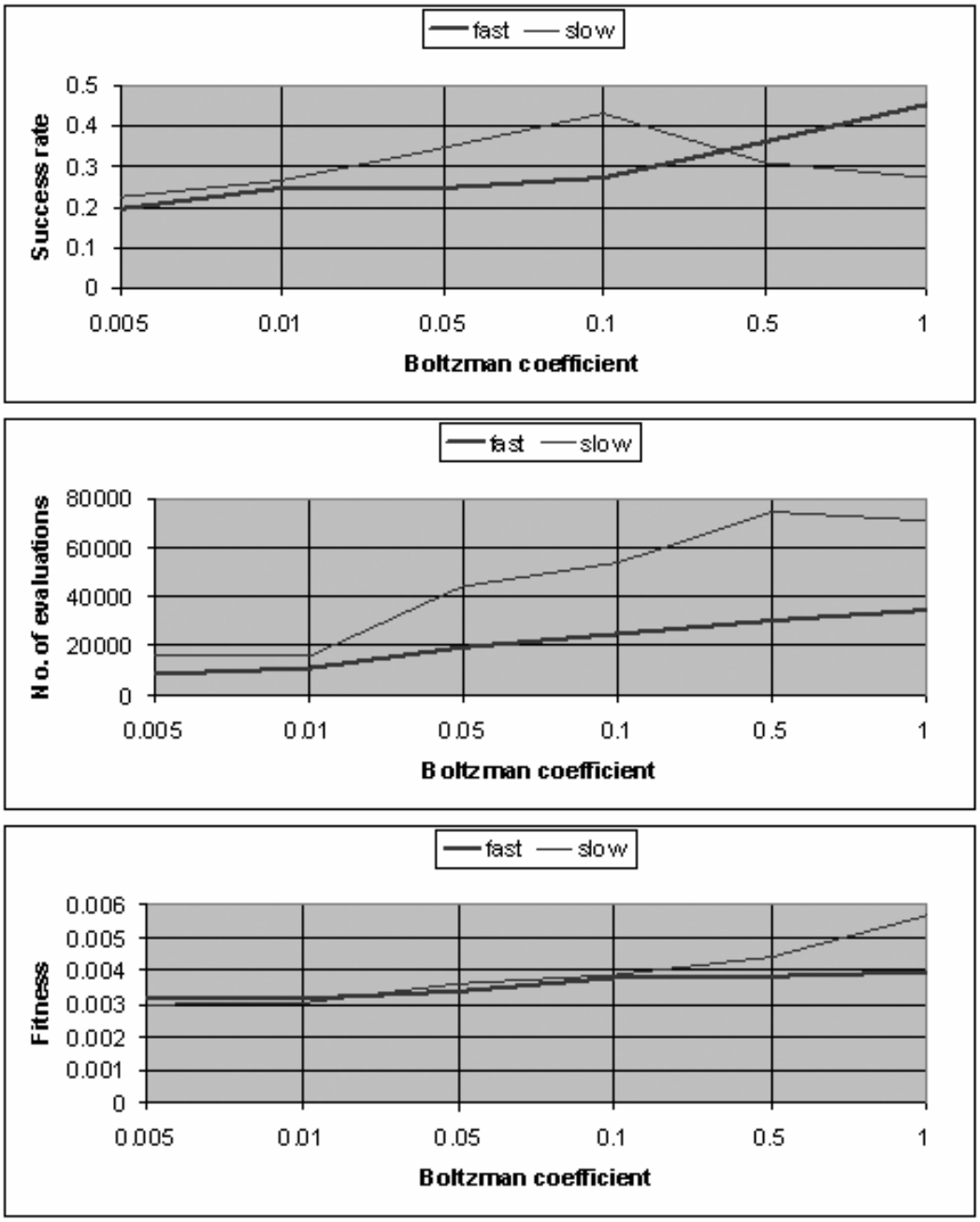

Figure 4.13: Performance characteristics for two cooling schedules in the Simulated annealing run

The mean value of the number of function evaluations over all successful solutions for the slow schedule is significantly higher than the mean value for the fast schedule; it

grows much faster as well, up to the point corresponding to $k = 0.5$. The mean values for the point $k = 0.005$ equal 16292 and 8676, for the slow and fast schedules, respectively. The mean values for the point $k = 1.0$ equal 70829 and 34270, for the slow and fast schedules, respectively. The mean values of the best solution F, over all successful solutions are very close for the both, slow and fast schedules, up to the point corresponding to $k = 0.1$; they slowly degrade, as the value of Boltzmann's constant increases. At the point k = 0.005, the mean values of the best solution equal $F = 0.00298$ and $F = 0.00348$, for the slow and fast schedules, respectively. After the point k = 0.1, the quality of the best solution obtained with the slow schedule is quickly degrading, while the quality of the best solution obtained with the fast schedule stays almost at the same level. At the point k = 1.0, the mean values of the best solution equal $F = 0.00568$ and $F = 0.00398$, for the slow and fast schedules, respectively.

The Multi-start algorithm with DSM uses the same distribution of the initial candidate solutions, as the parallel Simulated annealing. For each starting point, the initial DSM simplex is built using two different configurations:

- small configuration, with the simplex vertices drawn at random from the (± 5%) range about the starting point;
- large configuration, with the simplex vertices drawn at random from the (± 10%) range about the starting point.

For the small configuration, the algorithm starts with the mean value of the best solutions $F = 0.49633$, over 133 best initial simplex vertices. The total number of function

evaluations over all local search instances is 6087. The best solution $F = 0.04493$, with the parameter vector $\boldsymbol{V} = \{247, 134, 3.13, 4.87, 4.16\}$ is obtained after 57 function evaluations. The mean value of the best solutions over 133 best final simplex vertices is $F = 0.20276$, a 59% improvement over the initial simplexes. For the large configuration, the algorithm starts with the mean value of the best solutions $F = 0.33163$, over 133 best initial simplex vertices. The total number of function evaluations over all local search instances is 7094. The best solution $F = 0.03868$, with the parameter vector $\boldsymbol{V} = \{251, 157, 3.13, 4.78, 3.25\}$ is obtained after 56 function evaluations. The mean value of the best solutions over 133 best final simplex vertices is $F = 0.18347$, a 45% improvement over the initial simplexes. Although the algorithm achieves significant improvement of the value of the objective function F, it fails to find the global optimum solution, i.e., the success rate is zero, for the both, small and large configurations.

The same classical Genetic algorithm with elitist reproduction, single-point crossover, and random mutation is applied to the images, as described in section 4.2 of this Chapter. The initial chromosome population size is $P = 133$; the 20% fraction of the fitter chromosomes are copied directly into the next generation; the mutants form 3% of the chromosome population. The algorithm is augmented with the two-phase cyclic local search discussed in section 4.3.5 of this Chapter. Hybrid Evolutionary Algorithm starts with the same distribution of the initial candidate solutions, as parallel Simulated annealing and Multi-start. The algorithm is able to find the global optimum solution $F = 0.00220$, with the parameter vector $\boldsymbol{V} = \{150, 212, 1.56, 4.00, 2.06\}$, after 12 global iterations and 5148 function evaluations, with 1597 (31%) evaluations attributed to the

local search. Figure 4.14 shows the final optimum position of the object image in the scene found by the algorithm.

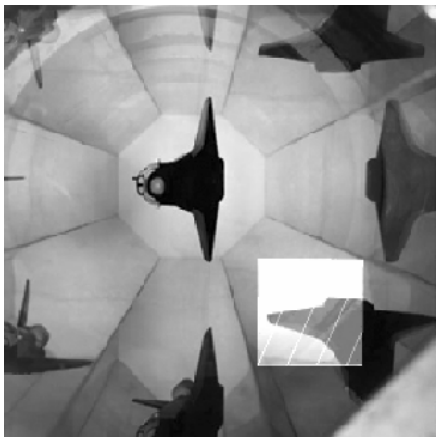

Figure 4.14: Optimum position of the object template found by the HEA with the two-phase cyclic local search

	Parallel SA	**Multi-start**	**HEA**
Success rate (min / max)	0.271 / 0.451	0.0 / 0.0	1.0 / 1.0
Evaluations (min / max)	8676 / 70829	6087 / 7094	5148 / 5148
Best fitness (min / max)	0.00298 / 0.00568	0.03868/0.04493	0.0022/0.0022

Table 4.5: Comparative performance of stochastic optimization methods

Table 4.5 summarizes the findings of the experiments. As one can see, Hybrid Evolutionary Algorithm augmented with the proposed two-phase cyclic local search procedure, has significant advantage over the other two methods, across all evaluation

criteria. It provides the best optimal solution $F = 0.00220$, with the probability (success rate) 1.0, and with the minimum number of function evaluations 5148. The values of the components of the best parameter vector $\boldsymbol{V} = \{150, 212, 1.56, 4.00, 2.06\}$ are equal or very close to the exact optimal values $\boldsymbol{V}^* = \{150, 212, 1.57, 4.14, 2.07\}$.

4.4. Increasing Efficiency of Mutation

This section discusses two techniques aimed at increasing the efficiency of mutation in EAs, namely, adaptive size of mutation pool (section 4.4.1), and mutation with memory (section 4.4.2).

4.4.1. Adaptive Size of Mutation Pool and Terminating Criteria

The classical evolutionary strategy relying on the basic genetic operators of crossover, low-rate mutation, and selection shows slow convergence for imaging optimization problems, as discussed in section 4.2 of this Chapter. The iterative process can easily fall into a local minimum, and stay there for an uncertain amount of time. The basic operations of elitist reproduction and crossover in this case can make the situation even worse. Reproduction will carry the false solution over to the next generations. Crossover will gradually increase the number of offspring having the same or similar genes that the fitter individuals have. The offspring cloning the fitter solutions will be increasingly replacing other “non-fit” chromosomes. Mutation is the operator that can help the search crawl out from the local minimum. However, if the mutation rate is low (a few percent, or even a fraction of a percent, as in many evolutionary schemes), it might take a considerable amount of time for mutation to come across a fitter solution.

A reasonable remedy that can prevent the evolutionary process from stalling at a local minimum is the increasing amount of mutation, in order to maintain a satisfactory degree of diversity in the chromosome pool. The size of the mutation pool can be controlled by the rate at which the gradient of the fitness function is decreasing from iteration to iteration. In order to establish the relationship between the convergence rate of the fitness function and the size of the mutation pool, a weighted error factor F_{we} is introduced as follows. At every iteration, the sum of fitness gradients in the replication pool is computed and weighted over the minimum fitness value at the iteration, as follows:

$$F_{\text{we}} = \frac{\sum_{R} \nabla F_i}{F_{\min}}, \tag{4.7}$$

where ∇F_i is the fitness gradient for the i-th chromosome in the current replication pool, $F_{\min}$ is the minimum fitness value for the current iteration, and R is the size of the replication pool. If the value of F_{we0} for iteration 0 (i.e., the initial iteration) is chosen as the base value, then the relative value of the weighted error factor $F_{\text{wer}k}$ for every subsequent iteration k is computed, as follows:

$$F_{\text{wer}k} = F_{\text{we0}} / F_{\text{we}k}\ , \tag{4.8}$$

where $F_{\text{we}k}$ is the current absolute value of the weighted error factor.

The size of the mutation pool for the next generation has to be increased by the factor F_{wer}. As the fitness function is flattening between iteration, the value of F_{wer} and, consequently, the size of the mutation pool has to grow. The larger number of new individuals is introduced into the chromosome population, thus helping the algorithm

climb out from the local minimum point and continue its search for the global optimum solution.

As an alternative approach, the sum of fitness values in the entire replication pool can be used as the weight coefficient, to compute F_{ew}, as follows:

$$F_{\text{we}} = \frac{\sum_{R} \nabla F_i}{\sum_{R} F_i}, \tag{4.9}$$

where F_i is the fitness value of the i-th chromosome in the current replication pool. Computational results show, however, that computing F_{we} according to Formula (4.7) provides superior performance, over computing F_{we} according to Formula (4.9).

The effect of the adaptive mutation pool in the classical Evolutionary algorithm is tested on the same problem of aligning synthetic sensor readings, as is used in Chapter 1 – see Figure 1.2. The minimization problem is stated in the 3-dimensional parameter space, with the parameter vector $\boldsymbol{V} = \{DX, DY, \theta\}$, and the optimal solution $\boldsymbol{V}^* = \{91, 91, 2.78081\}$. The classical Genetic algorithm with elitist reproduction, single-point crossover, and random mutation is applied to the images, with the population size $P = 133$ and the elitist reproduction of 20% of the fitter chromosomes. The weighted error factor F_{we} is computed according to Formula (4.7), and applied to the procedure, in order to control the size of the mutation pool.

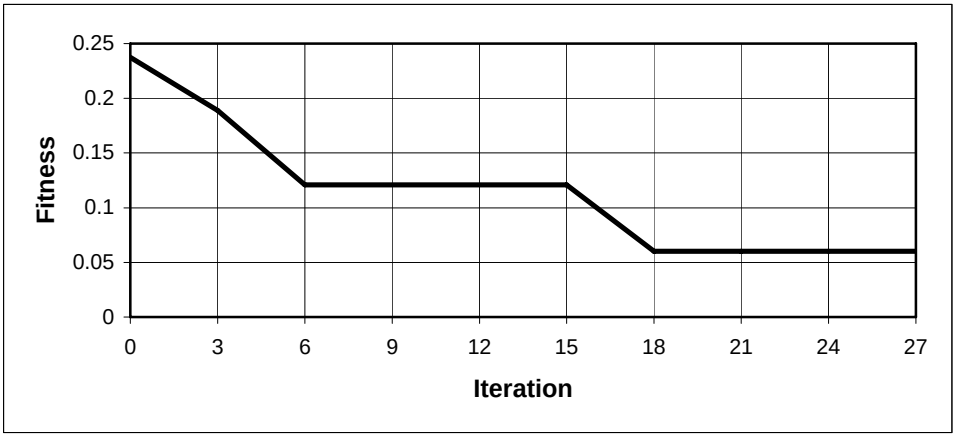

Figure 4.15: Performance of the elitist EA with the adaptive size of mutation pool, and the weighted error computed according to Formula (4.7)

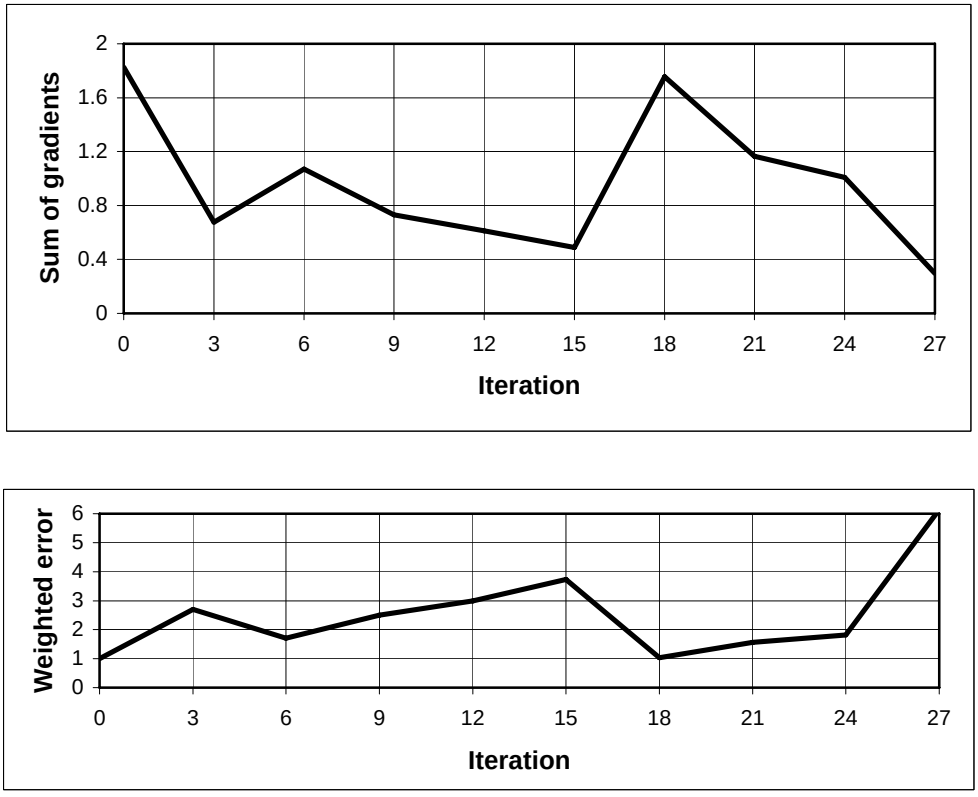

Figure 4.16: Sum of the fitness gradients (top), and the weighted error F_{wer} computed according to Formula (4.7) (bottom)

As in the original version of the elitist algorithm, the evolutionary process goes through a local minimum in iterations 6 – 15 (see Figure 4.15), with the parameter vector $\boldsymbol{V}$ = {109, 88, 2.78081}. However, the relative value of the weighted error F_{wer} does not change significantly: it has maximum value F_{wer} = 3.73302 at iteration 15, as compared to the initial value F_{wer} = 1.0 (see Figure 4.16). The value of F_{wer} grows exponentially from iteration 27, with the sum of the fitness gradients falling to 0. This behavior can serve as an indication that the global minimum point has been reached, and the search can be terminated, with the optimum parameter vector $\boldsymbol{V}$ = {95, 88, 2.78081}, minimum fitness value F = 0.06028, and the weighted Euclidean distance from the exact solution δ = 0.056.

These results provide experimental support for using the relative weighted error factor as a potential terminating criterion of the search. As opposed to monitoring the overall convergence of fitness function between iterations, the relative weighted error factor F_{wer} can be computed and compared at every iteration. When the factor begins to grow exponentially, the search can be terminated, because the global minimum solution has been found. Although this strategy provides the best final result, it might be too restrictive and computationally expensive: often the exact solution is not needed, and its approximation could suffice. The requirement of finding the point with the exponential growth of the factor F_{wer} can be relaxed, and a reasonable threshold value can be accepted. If F_{wer} exceeds the threshold, the found solution is considered to be satisfactory, and the search can be terminated.

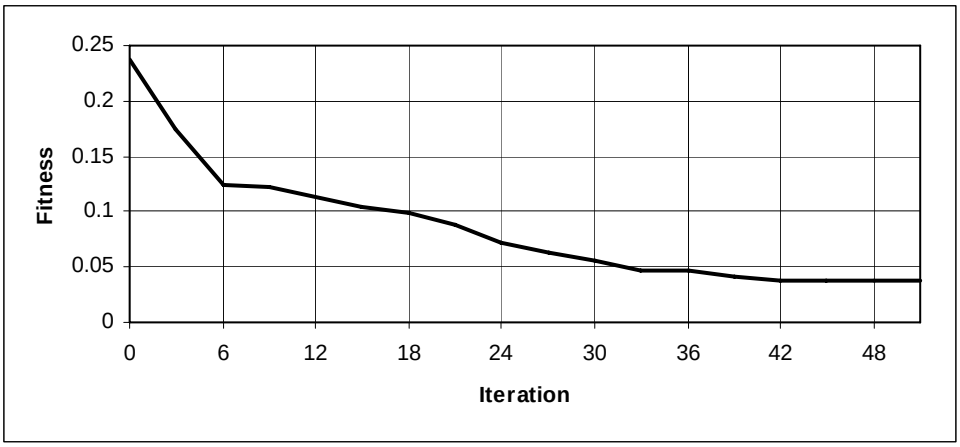

Figure 4.17: Performance of the elitist EA with the adaptive size of mutation pool, and the weighted error computed according to Formula (4.9)

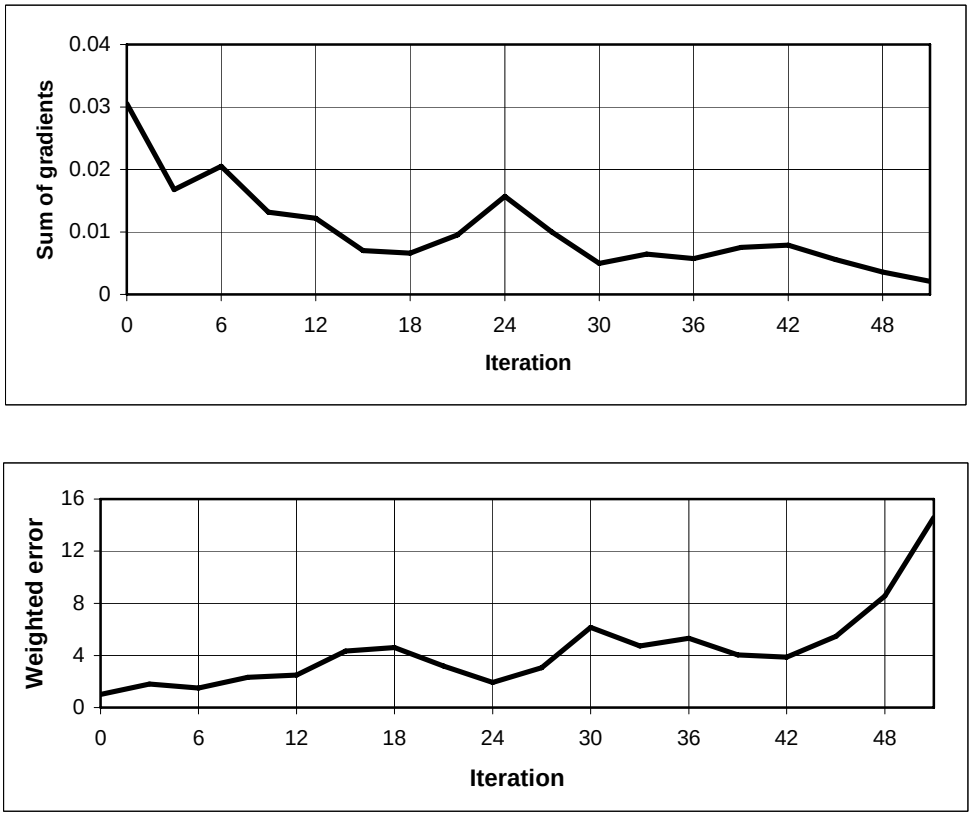

Figure 4.18: Sum of the fitness gradients (top), and the weighted error F_{wer} computed according to Formula (4.9) (bottom)

Another alternative to computing the weighted error factor with Formula (4.7), is to use the sum of the fitness values over the entire replication pool as the weight coefficient, as shown in Formula (4.9). Figures 4.17 and 4.18 show, however, that this approach requires a larger number of iterations, in order to reach the final optimum point. The factor F_{wer} grows slowly until it reaches iteration 51, where it begins to grow exponentially indicating that the optimum solution has been found. Although a satisfactory parameter vector $\boldsymbol{V} = \{89, 94, 2.74396\}$, with the minimum fitness value $F = 0.04702$, is found in iteration 33, the search cannot be stopped until after iteration 51. This result shows that using the minimum fitness value as the weighted factor, as stated in Formula (4.7), provides better performance than using the sum of the fitness values, as in Formula (4.9).

The attempt to extend the computation of the weighted error factor F_{we} to the entire chromosome pool, instead of its replication part, does not provide satisfactory results (see Figures 4.19 and 4.20). The relative factor F_{wer} can no longer be considered a monotonic function of the fitness value: it has three extreme values between iterations 0 and 48. Only one of them (at iteration 48) corresponds to the global minimum point, with the parameter vector $\boldsymbol{V} = \{91, 93, 2.75134\}$, and fitness value $F = 0.038$. Therefore, computing the factor F_{we} over the entire chromosome pool cannot provide a good indication as to when the optimum solution is found, and the search can be terminated.

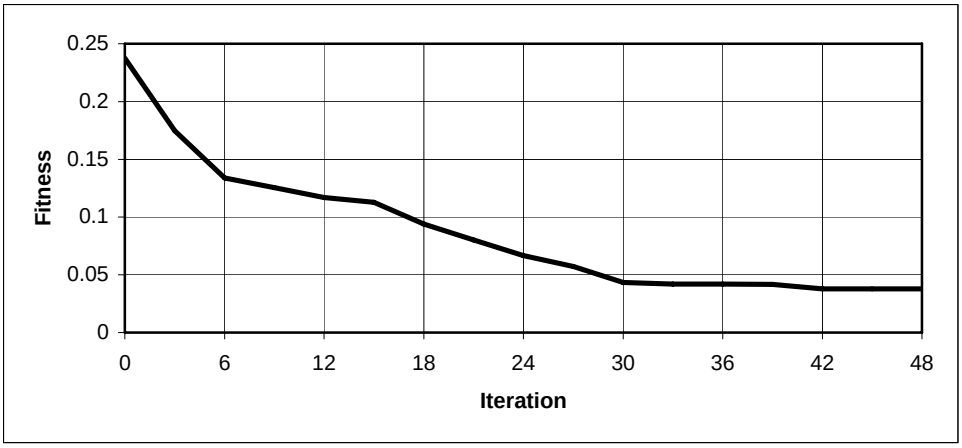

Figure 4.19: Performance of the elitist EA with the adaptive size of mutation pool, and the weighted error computed over the entire population

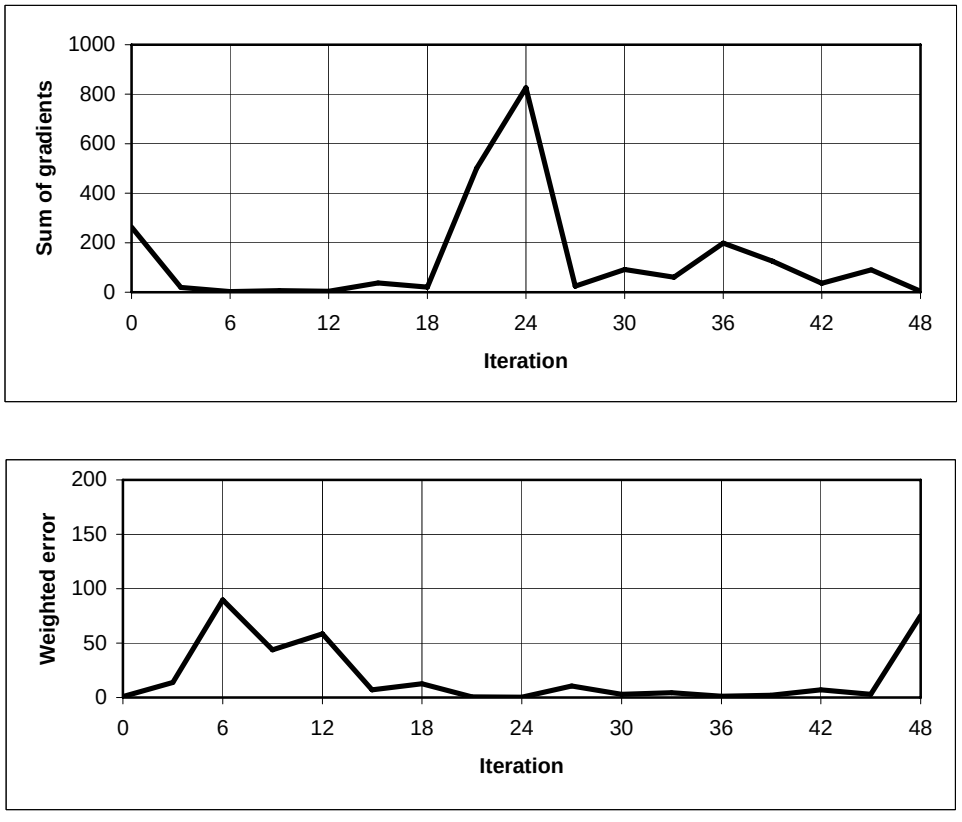

Figure 4.20: Sum of the fitness gradients (top), and the weighted error F_{wer} computed over the entire population (bottom)

The results of the experimental study aimed at evaluating the possibility of using adaptive size of mutation pool as means to accelerate the convergence of the classical Evolutionary algorithm, and to indicate the terminating condition, show that the best strategy is to use the relative weighted error factor F_{wer} computed according to Formulae (4.7) and (4.8), in combination with the adaptive size of mutation pool.

4.4.2. Mutation with Memory

The main objective of mutation is to diversify the chromosome population in such a way that different areas of the search space can have a fair chance to participate in forming new generations. The classical low-rate operation of mutation relies on a purely random selection of new individuals from the entire search space. In reality, however, some of the areas are better represented in the population than others. The inequality existing in the chromosome population is exacerbated when the following three techniques are used, in order to improve the EAs performance:

- elitism that directly copies fitter chromosomes into the next generation;
- increasing mutation rate that helps diversify the genetic material in the population;
- local search that speeds up the EAs convergence, and refines the quality of the fitter solutions.

The areas where the fitter chromosomes reside will be repeatedly re-visited in the chromosome pool, as the result of reproduction; intensified mutation; and local search and correction. If the uniform probability distribution is used to draw the mutants, the

areas that already have a relatively high representation will maintain, or even increase their domination in the population. Consequently, some areas that might potentially contain the global optimum will not be explored fully enough to find this optimum, while areas containing local optima will be over-represented.

In order to keep track of the fair usage of the search space, a bookkeeping mechanism is implemented as followed. The entire search space (domain D) is divided into a finite number of segments d_{nm}, such that

$$D = \sum_{n=1}^{N} \sum_{m=1}^{M_n} d_{nm} \ , \tag{4.10}$$

where N is the number of dimensions of the search space, and M_n is the number of segments in the n-th dimension.

A hit function $h_{nm}(t)$ is defined for a segment d_{nm} as a function of time t, as follows:

$$h_{nm}(d_{nm},t_k) = h_{nm}(d_{nm},t_{k-1}) + u_{nm,k} \ ,$$

$$h_{nm}(d_{nm},0) = 0 \ , \tag{4.11}$$

where t_{k-1} and t_k are the two consecutive time intervals, and $u_{nm,k}$ is the rate of usage of the parameter values $V_{nm} \in d_{nm}$ during all evolutionary operations of local search, crossover, and mutation, at the time interval t_k.

Another N-dimensional domain $R(t_k)$ is defined at every time interval t_k, as a set of ordered range segments, as follows:

$$R(t_k) = \sum_{i=1}^{N} \sum_{j=1}^{J_i} r_{ij}(t_k) \, , \tag{4.12}$$

where J_i is the number of the range segments for the i-th dimension, and the range segment $r_{ij}(t_k)$ is computed as

$$r_{ij}(t_k) = \frac{\max h_{iq}(d_{iq}, t_k) - \min h_{iq}(d_{iq}, t_k)}{J_i}. \tag{4.13}$$

Before mutation takes place at the time interval t_k, a new mapping $M(t_k)$ from the search space D into the domain $R(t_k)$ has to be found, as follows:

$$M(t_k) : D \rightarrow R(t_k) \text{ , such that } d_{nq} \rightarrow r_{np}(t_k) \quad \text{if } r_{np\text{-}1}(t_k) < h_{nq}(d_{nq}, t_k) < r_{np}(t_k) \tag{4.14}$$

The operation of mutation is performed at random only on the range segments whose values do not exceed some pre-set threshold $R_n(t_k)$. These segments correspond to the areas of the search space that are located away from the reproduction pool, and have been least visited by the search procedure. The technique insures that the subspaces that have been least represented up to date in the chromosome population, will have a higher probability to be re-visited during the later stages of the evolutionary search.

The evolutionary procedure now incorporates two opposite processes that take place simultaneously in time. Operations of reproduction; local search and correction; and crossover increase the frequency of re-visiting the areas that surround local (or, possibly the global) minimum points in the reproduction pool. Mutation with memory, on the contrary, draws new individuals from the areas that have been less frequently visited, and are located outside the densely populated spots in the reproduction pool. Mutation attempts to explore all poorly represented areas, in order to eliminate the possibility of overlooking a better (or the best) solution. Together, these two processes form two

opposite but complementary trends, concentration and expansion, increasing the probability of finding the global optimum solution, no matter where it is located, in the vicinity of the current reproduction pool, or at a far distance from it.

4.5. Comparison of Modified and Classical Versions of Evolutionary Algorithm

In order to evaluate the performance gain of the proposed modified version of the evolutionary strategy that includes the two-phase cyclic local random/DSM search and mutation with memory, comparative experiments are conducted using the modified algorithm and the classical elitist EA strategy, The same test sets of the truck and wing images are used here, as the test sets discussed in section 4.2 of this Chapter.

Charts in Figures 4.21 and 4.22 show the performance of the modified Evolutionary algorithm, for the both sets of the test images. The best solution for the truck image, with the parameters $\boldsymbol{V}$ = {24, 141, 0.76054, 2.04093} and the minimum fitness value F = 0.00370 is found in iteration 11 – see Figure 4.21. The total number of fitness evaluations is 3421. The best parameter set for the wing image is found in iteration 17, and requires 5467 fitness evaluations – see Figure 4.22. It has parameters $\boldsymbol{V}$ = {170, 241, 1.58207, 4.07248} and the minimum fitness value F = 0.00246. Figure 4.23 shows the matched images obtained with the modified EA procedure. Table 4.6 summarizes the comparative computational performance of the modified and classical elitist EA models.

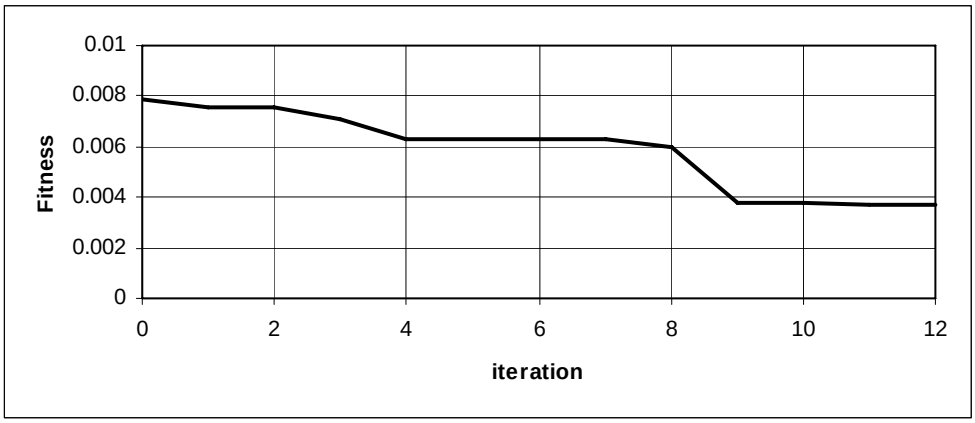

Figure 4.21: Performance of the modified EA for the truck image

Figure 4.22: Performance of the modified EA for the wing image

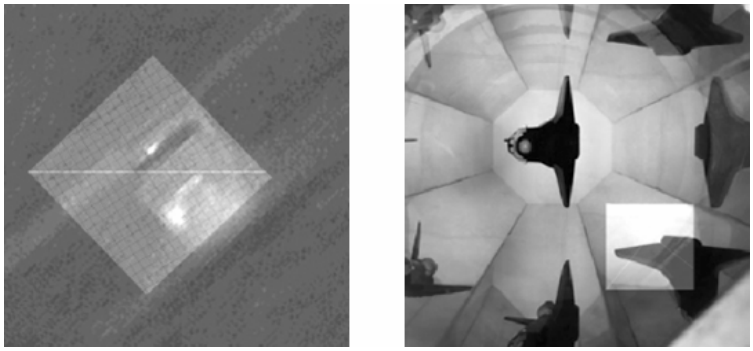

Figure 4.23: Matched images of the truck (left) and the wing (right) obtained with the modified HEA

	Truck			Wing		
	Classical approach	**Modified approach**	**Exact solution**	**Classical approach**	**Modified approach**	**Exact solution**
Parameters:						
DX	23	24	--	166	170	170
DY	158	141	--	245	241	240
θ	0.89664	0.76054	0.7854	1.56509	1.58207	1.5734
S	2.03323	2.04093	--	3.6368	4.07248	4.14
Fitness *F*	0.00464	0.00370	--	0.0062	0.00246	0.00215
Iterations	391	11	--	77	17	--
Evaluations	44213	3421	--	8731	5467	--

Table 4.6: Comparative performance of the classical and modified EA models

An interesting example of changes occurred in the parameter and fitness values between iterations, for the fittest chromosome of the wing image, during one run of the modified EA model is shown in Figure 4.24. Also shown, are operations that are responsible for the changes. The total number of fitness evaluations for the fittest chromosome is 175, including 91 during the random search, and 84 during the DSM refinement. As one can see, random selection of trial points from the neighborhood of the fittest chromosome improves the best solution in the same manner as the classic crossover does. The refinement of the solution with DSM mostly affects the later stages of the evolutionary search. Interestingly enough, the correction of other chromosomes from the reproduction pool contributes to the success of the entire search. For example, the random correction

of the chromosome 4 followed by the random correction of the chromosome 0 contributes in iteration 6. The DSM refinement of the chromosome 1 in iteration 17 produces the best global solution for the entire search. This result clearly indicates that using local search only for the final solution, or a subset of the final solutions obtained at the end of the global evolutionary search, is not as efficient as using local search during all intermediate stages of the search.

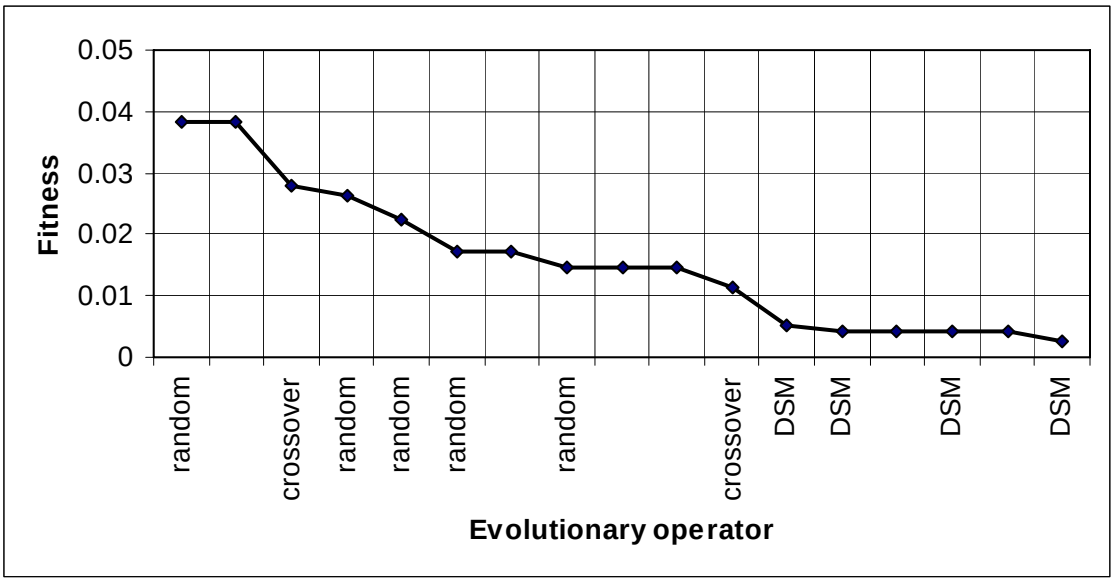

Figure 4.24: Example of the changes of the best fitness value

The total number of local fitness evaluations for the wing image is 1381, including 1230 random trials, and 151 DSM refinements. This is a 25.26% fraction of the total number of fitness evaluations required for the entire solution. The number of the successful random trials is 96; 81 trials involve only 4 neighbors located inside a hypersphere with just one radius away from the initial chromosome. Therefore, a careful choice of the initial radius

of the explored neighborhood can significantly reduce the number of the probed neighbors and, consequently, the total number of trial points during the random search.

In summary, the computational experiments with two test sets show that the proposed modification of the classical evolutionary procedure with the two-phase cyclic local random/DSM search and mutation with memory, significantly increases the convergence rate simultaneously reducing the required number of fitness evaluations. The classical evolutionary procedure produces the best solution $F = 0.00464$ after 391 iterations, with 44213 fitness evaluations, for the truck image, while the modified procedure reaches the best point $F = 0.00370$ after 11 iterations and 3421 fitness evaluations. Similar results are obtained for the wing image. The best solution after applying the classical procedure is $F = 0.0062$; it requires 77 iterations and 8731 fitness evaluations. The modified procedure reduces the number of iterations to 17, with 5467 fitness evaluations, while gaining a much better final result of $F = 0.00246$.

4.6. Selective Fitness Evaluation as Problem-Specific Knowledge

When Evolutionary algorithms are applied to optimization problem, the computational time T_J required for performing J evolutionary iterations is

$$T_J = \sum_{j=1}^{J} \sum_{g=1}^{G_j} E_{jg} t_e + O(J) + O(G), \tag{4.15}$$

where G_j is the number of chromosomes at iteration j, E_{jg} is the number of fitness evaluations for the chromosome g at iteration j, and t_e is the time required for one fitness evaluation. The terms $O(J)$ and $O(G)$ are attributed to various auxiliary operations. The overall computational efficiency of the algorithm, therefore, can be potentially improved

by reducing the values of the parameters J, G_j, E_{jg}, and t_e of the iterative process. The first three parameters depend on the overall strategy of the evolutionary search. The reduction of the fourth parameter, the time t_e required for one fitness evaluation, is discussed in this section.

In the statement of image mapping as an optimization problem, every operation of fitness evaluation requires $N \times M$ operations of geometric transformation applied to $N \times M$–pixel image, and $N \times M$ operations of pixel-wise comparison of two images. Since the number of function evaluations is usually large (in the range of a few thousands), even a small reduction of the time t_e would result in significant reduction of the total computational cost of finding a solution. In particular, if one can reduce the number of pixels participating in the operations of transformation and comparison, then the fitness evaluation will require only $N' \times M'$ operations, where $N' < N$, or $M' < M$, or both. The reduction of the time required for one fitness evaluation can be estimated using the area reduction factor Φ_A defined as

$$\Phi_A = \frac{N' \times M'}{N \times M} \ , \tag{4.16}$$

where $\Phi_A < 1$. Since the time t_e required for one evaluation is constant, Formula (4.15) can be written as

$$T_J = t_e * E_{JG} + O(J) + O(G) \ , \tag{4.17}$$

where E_{JG} is the total number of fitness evaluations. Using the area reduction factor Φ_A, we can now define the effective number of fitness evaluations as

$$E_E = \Phi_A * E_{JG} \ . \tag{4.18}$$

This section compares performance of two algorithms that can be potentially used to reduce the fitness evaluation time – the fractal encoding of the image and a simple image segmentation technique. Both algorithms aim at selective reducing the number of pixels participating in the fitness evaluation.

4.6.1. Selective Fitness Evaluation with Fractal Encoding

One of the techniques that can be utilized to reduce the number of pixels participating in the fitness evaluation is fractal encoding commonly used in image compression [174]. Fractal image encoding exploits the idea of a partitioned iterated function system. Encoding starts with representing the image as two independent sets, a set of domain cells D_i, and a set of range cells R_j. The algorithm then attempts to find a set of contractive affine transformations M_{ij} from D_i into R_j. The transformations defined by their corresponding parameters constitute the desired compression of the original image. The process of fractal encoding is used here to reduce the fitness evaluation time, in the following way.

Step 1: The image is partitioned into a set of domain cells D_i of regular rectangular shape. For every domain cell, a vector of features is extracted containing the following information about the distribution of gray values over the domain pixels:

- standard deviation σ of the pixel values;
- skewness, defined as the normalized sum of the cubes of the differences between the pixel values and their mean;
- neighbor contrast, measuring the average difference between the adjacent pixels;

- coefficient β measuring the change of the pixel values, in relation to the center of the cell;
- maximum gradient of the pixel values.

Step 2: The image is partitioned into a set of range cells R_j using the adaptive quadtree partitioning scheme. The range cells are smaller than the domain cells, in order for the mapping M_{ij} to be contracting. The vector of features is extracted for every range cell.

Step 3: Domain D_i, and a corresponding affine transformation M_{ij} are found for every range cell R_j, such that the feature vectors of D_i and R_j fit, with a specified error tolerance. The transformation M_{ij} includes translation, rotation, and shrinking of the domain cell. If the fit is not sufficient, the range cell is subdivided into smaller cells using quadtree partitioning, and search for the best fit continues. Information about the range cell, the corresponding best domain, and the parameters of the affine transformation is stored for further processing.

Step 4: Once information about all range cells has been computed and stored, the smallest range cell R_{min} is found. The cell threshold T_R is set, based on the size of R_{min}. The list of the range cells obtained in step 3 is processed, and all cells exceeding the threshold T_R are discarded. The cells that are left in the list are used for fitness evaluation during the iterations of the evolutionary procedure.

Steps 1 - 3 are the standard steps of fractal encoding commonly used in image compression [174]. Step 4 is based on the observation that the encoding algorithm sorts out the areas containing a larger amount of information about the image, from the areas containing less information. The large-size range cells obtained in step 3 of the algorithm have the homogeneous distribution of their gray values; therefore, they contain fewer details about the image. On the contrary, smaller cells contain the large number of the interesting details about the image, which makes them more useful and important for image comparison during the mapping process. The number of domain cells, the depth of the adaptive range partitioning, and the error tolerance can significantly affect the partitioning results and the number of cells participating in the fitness evaluation during the algorithm run.

4.6.2. Selective Fitness Evaluation with Image Segmentation

A simple and inexpensive alternative to fractal encoding is a simple segmentation technique that attempts to find the borders between homogeneous areas of the image, in order to discard most of the pixels within these areas, and to leave only sufficient representative layer along the borders; this layer will carry enough information about the area texture and its averaged gray value. The algorithm processes the image in a row-wise manner, as follows.

Step 1: For every row, a sliding window is built that consists of three successive pixel segments, Ω_n, Ω_{n+1} and Ω_{n+2}. The segments are obtained in such a way, that the

difference of their average gray values (AGV) exceeds some pre-set gray value tolerance (GVT):

$$| \mathrm{AGV}_{n+1} - \mathrm{AGV}_n | \geq \mathrm{GVT} , \quad | \mathrm{AGV}_{n+2} - \mathrm{AGV}_{n+1} | \geq \mathrm{GVT} . \tag{4.19}$$

Step 2: If the segment Ω_{n+1} is small, i.e., its length is below some pre-set threshold (here, the threshold of 1 pixel is used), it is treated as noise. The gray values of its adjacent neighbors are compared with the average gray values of the other two segments, Ω_n and Ω_{n+2}. If the majority of the neighbors have their gray values equal to AGV_n or AGV_{n+2} (with the tolerance GVT), the segment Ω_{n+1} is merged with the corresponding segment Ω_n or Ω_{n+2}. If $\mathrm{AGV}_n = \mathrm{AGV}_{n+2}$, then all three segments Ω_n, Ω_{n+1}, and Ω_{n+2} are merged.

Step 3: The segment Ω_n is now considered as a candidate for discarding from the image. If the length of Ω_n exceeds some pre-set minimum value $L_{\min}$, the segment is discarded, and its pixels do not participate in the later operations of the fitness evaluation. While discarding, the segment's edge pixels are kept intact, in order to provide a sufficient area on the border between the segments.

The value of the area reduction factor Φ_A can be controlled by varying two parameters, the gray value tolerance GVT, and the minimum length of the discarded segment $L_{\min}$. The ratio of these parameters (GVT / $L_{\min}$) is referred to as the tolerance factor.

4.6.3. Computational Experiments with Selective Fitness Evaluation

Two sets of grayscale images are tested on a 4-dimensional optimization problem, with the vector of parameters $\boldsymbol{V} = \{DX, DY, \theta, S\}$: the images of a truck and an aircraft wing – see Figure 4.1, section 4.2 of this Chapter. The first set includes two 286×286-pixel images of a truck, with unknown parameters *DX*, *DY* and *S*, and rotation $\theta = 0.7867$. The second set consists of two 290×290-pixel images of an aircraft wing, with the known parameter vector $\boldsymbol{V^*} = \{170, 240, 1.5734, 4.14\}$. The modified hybrid EA model with the two-phase cyclic local correction and mutation with memory is used. The selection mechanism is changed to a more efficient tournament selection, and the initial population seed is different from the seed used in section 4.5 of this Chapter.

Figure 4.25 shows the performance of HEA, in terms of the fitness value and number of fitness evaluations, in the case when the full *N*×*M*-pixel images are used for the fitness evaluation. The minimum solution for the truck is found after 5 iterations, with the parameter vector $\boldsymbol{V} = \{23, 135, 0.7250, 1.964\}$, and the total number of fitness evaluations 813. The minimum solution for the wing is found in iteration 8, with the parameter vector $\boldsymbol{V} = \{170, 239, 1.5303, 3.925\}$, and 1406 fitness evaluations.

For the fractal encoding, a two-level domain partitioning scheme is used. The first level has 8×8 = 64 domain cells, and the second level has 16×16 = 256 domain cells. The maximum quadtree depth for the range partitioning equals 5. According to the step 4 of the encoding algorithm described in section 4.6.1 of this Chapter, the reduction of the number of pixels participating in the fitness evaluation is achieved by discarding all range

cells that have a greater size than the size R_{min} corresponding to the level 5 of the quadtree. The variation of the error tolerance during the feature evaluation (referred to as the feature tolerance) results in different values of the area reduction factor Φ_A.

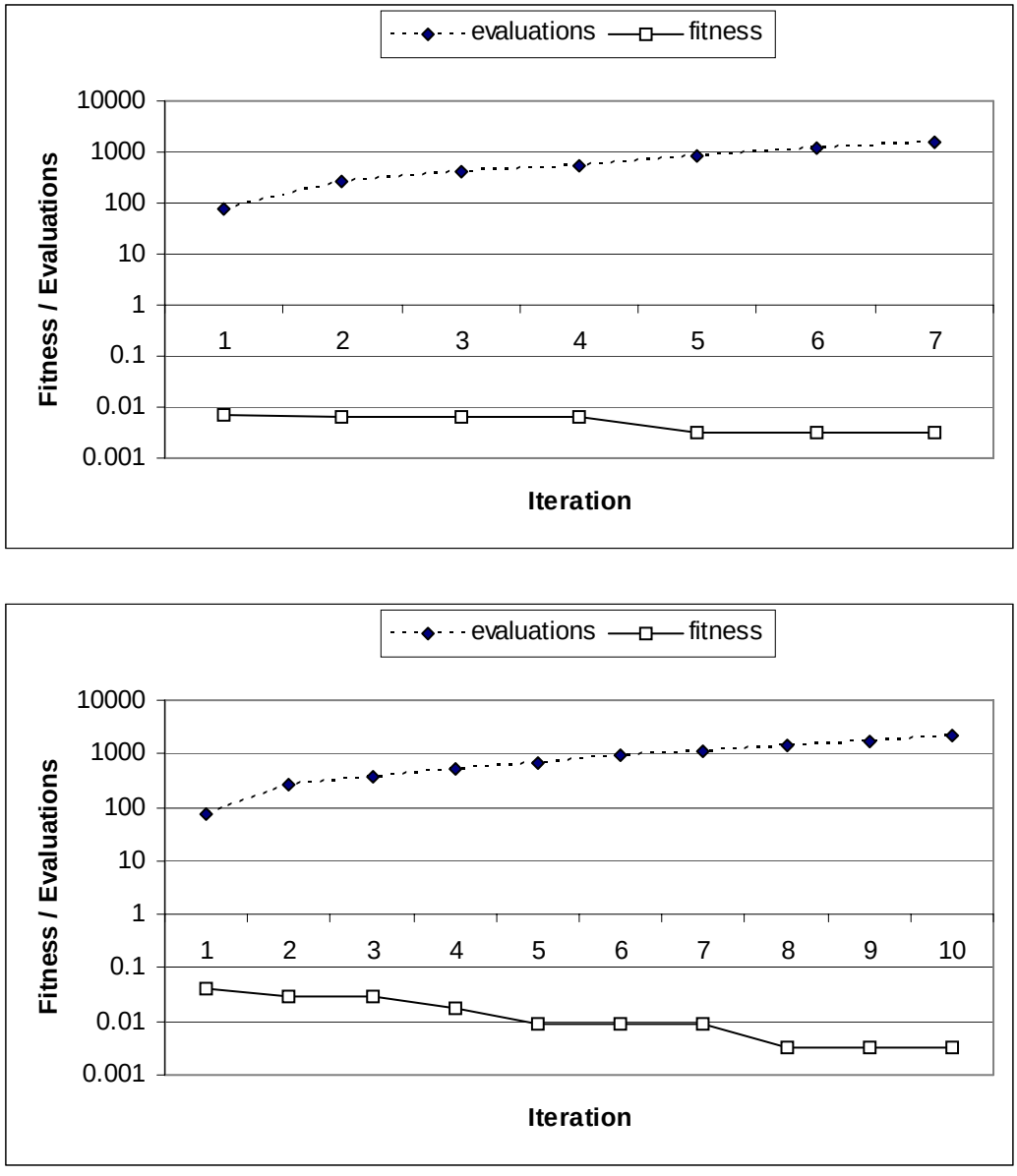

Figure 4.25: Performance of the algorithm with the full image evaluation: a truck (top) and a wing (bottom)

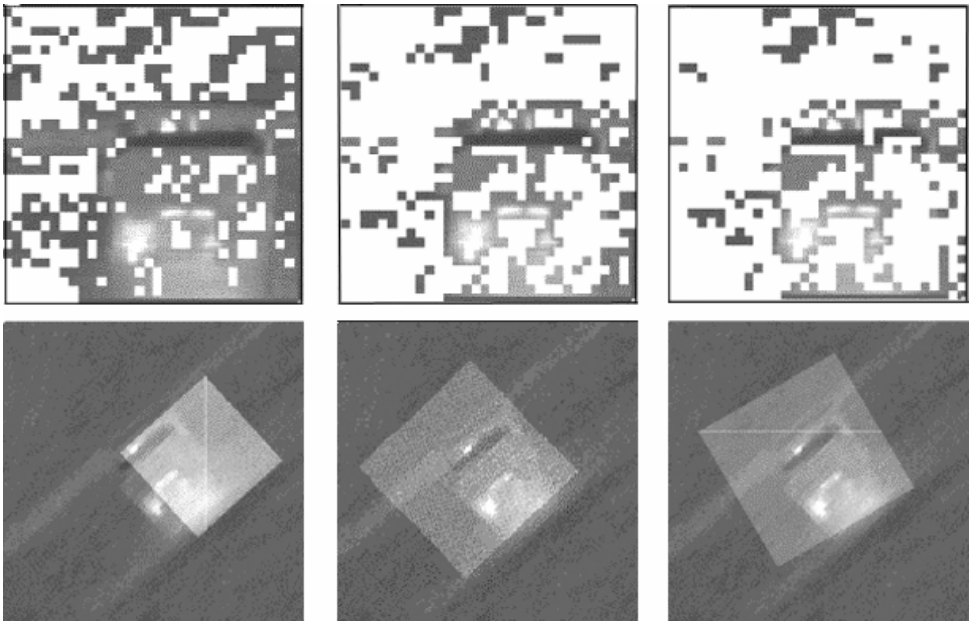

Figure 4.26: Fractal encoding (top) and mapping results (bottom) for the truck image

For the image of a truck, a range of feature tolerance between 0.025 and 0.1 is tested. The sample image reduction and mapping results are shown in Figure 4.26. Here, the values of the feature tolerance 0.025, $\Phi_A = 0.612$, $\theta = 5.4505$ (left column); feature tolerance 0.05, $\Phi_A = 0.322$, $\theta = 0.7633$ (central column); feature tolerance 0.1, $\Phi_A = 0.257$, $\theta = 0.5274$ (right column). The chart in Figure 4.27 shows how the area reduction factor Φ_A and the rotation angle θ change when the feature tolerance increases: while Φ_A monotonically decreases, the value of θ reveals a more interesting pattern. It falls from $\theta = 5.4505$, at the feature tolerance 0.025 and $\Phi_A = 0.612$ (not shown), down to $\theta = 0.1241$, at the feature tolerance 0.03 and $\Phi_A = 0.501$, then rises and comes very close ($\theta = 0.7633$) to the exact solution $\theta = 0.7867$, at the feature tolerance 0.05 and $\Phi_A = 0.322$. The angle slowly degrades to $\theta = 0.5274$, at the feature tolerance 0.1 and $\Phi_A = 0.257$. There is an optimal range of the area reduction factor (between $\Phi_A = 0.369$, and $\Phi_A = 0.302$ in Figure

4.27, corresponding to the range 0.04 – 0.06 of the feature tolerance), where the parameter values come very close to the exact solution.

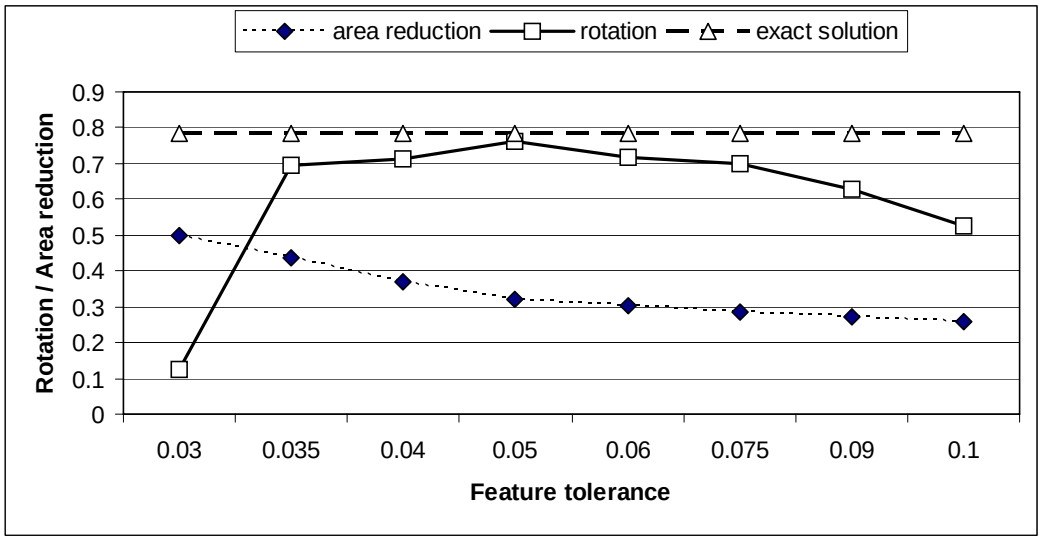

Figure 4.27: Area reduction factor Φ_A and rotation angle θ, as functions of the feature tolerance, for the fractal encoding of the truck image

Figure 4.28 shows the sample image reduction and mapping results for the wing image. The feature tolerance in this case varies from 0.005 to 0.05, with the corresponding variation of the area reduction factor between $\Phi_A = 0.749$ and $\Phi_A = 0.019$. In Figure 4.28, the values of feature tolerance 0.005, $\Phi_A = 0.749$, $\delta = 0.2$ (left column); feature tolerance 0.025, $\Phi_A = 0.091$, $\delta = 0.104$ (central column); feature tolerance 0.05, $\Phi_A = 0.019$, $\delta = 1.17$ (right column). As in the case of the truck, the value of Φ_A monotonically decreases with the increase of the feature tolerance – see Figure 4.29. The overall quality of the obtained parameter vector for the wing is estimated using the relative error δ defined as follows:

$$\delta = \sqrt{\left(\frac{\Delta DX}{DX}\right)^2 + \left(\frac{\Delta DY}{DY}\right)^2 + \left(\frac{\Delta \theta}{\theta}\right)^2 + \left(\frac{\Delta S}{S}\right)^2}, \tag{4.20}$$

where ΔDX, ΔDY, $\Delta\theta$, and ΔS are the differences between the exact and the approximate parameter values.

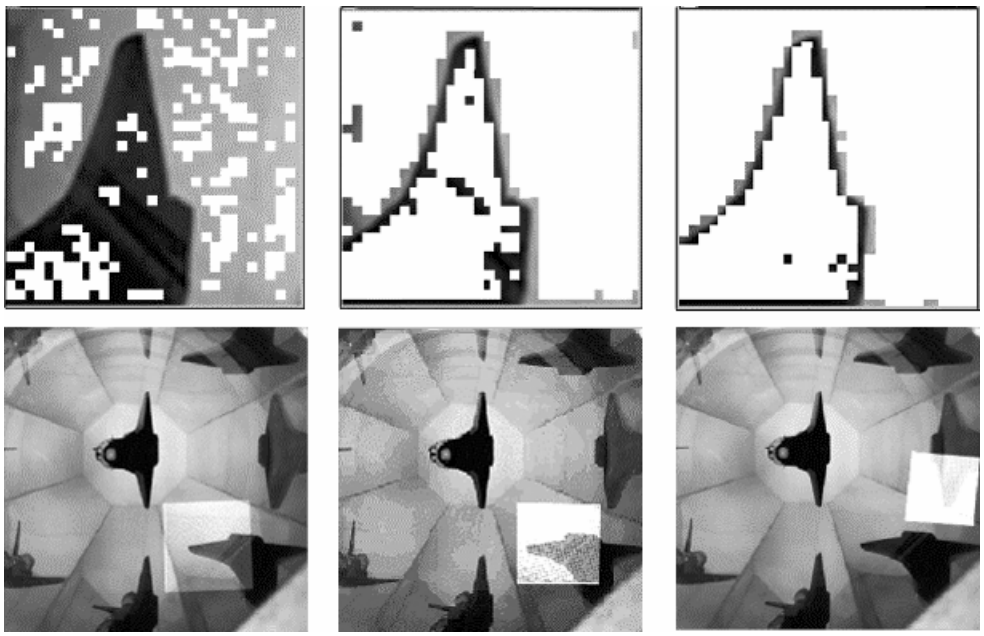

Figure 4.28: Fractal encoding (top) and mapping results (bottom) for the wing image

Unlike in the case of a truck, the value of the normalized error for a wing does not show a particular pattern. There are two points in Figure 4.29, where the trial solution comes very close to the exact optimum. The first point corresponds to the value of the feature tolerance 0.0075, with the parameter vector $\boldsymbol{V}$ = {170, 244, 1.6500, 4.0}, and the area reduction factor Φ_A = 0.496. The second good approximation point is found at the feature tolerance 0.025, with the parameter vector $\boldsymbol{V}$ = {168, 243, 1.556, 3.72}, and the area reduction factor Φ_A = 0.091.

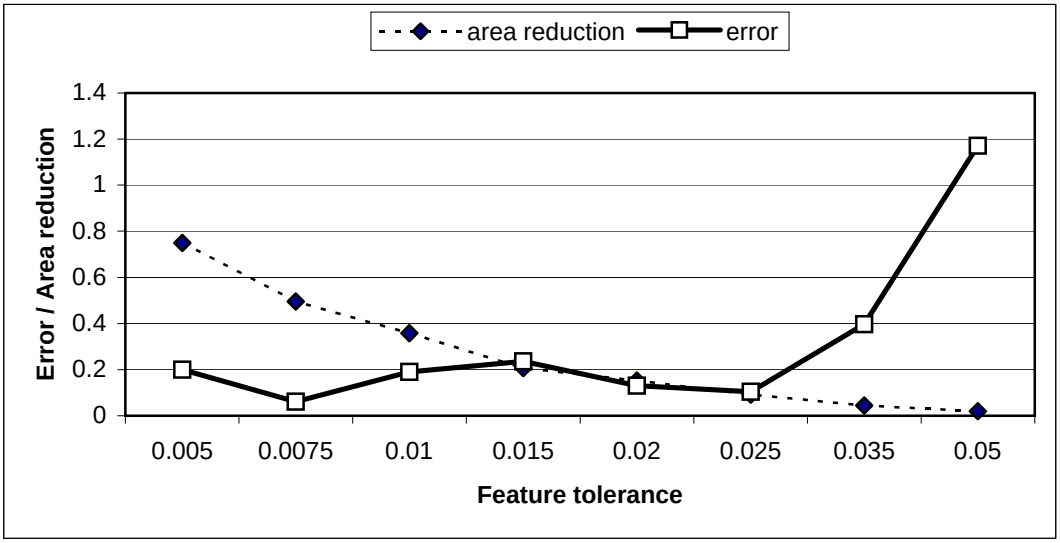

Figure 4.29: Area reduction factor Φ_A and the relative error δ, as functions of the feature tolerance, for the fractal encoding of the wing image

Similar experiments are conducted with the simple segmentation technique described in Section 4.6.2. In this case, there are two parameters that control the value of the area reduction factor – the gray value tolerance GVT, and the minimum length of the discarded segment $L_{\min}$. Sample image reduction and mapping results for a truck are presented in Figure 4.30. Here, the values of tolerance factor 0.1, $\Phi_A = 0.706$, $\theta = 0.1$ (left column); tolerance factor 0.8, $\Phi_A = 0.585$, $\theta = 0.75$ (central column); tolerance factor 1.6, $\Phi_A = 0.178$, $\theta = 0.4895$ (right column). The chart in Figure 4.31 shows the values of Φ_A and θ, as functions of the tolerance factor. In Figure 4.31, the value of the area reduction factor Φ_A decreases monotonically, as the tolerance factor (GVT / $L_{\min}$) increases. The value of the rotation angle does not show a particular pattern. It is very close to the exact solution at the point where the tolerance factor equals to 0.8, and the

area reduction factor Φ_A = 0.585; the value of θ at this point is 0.75. At the larger and smaller values of Φ_A, the value of θ significantly differs from the exact solution.

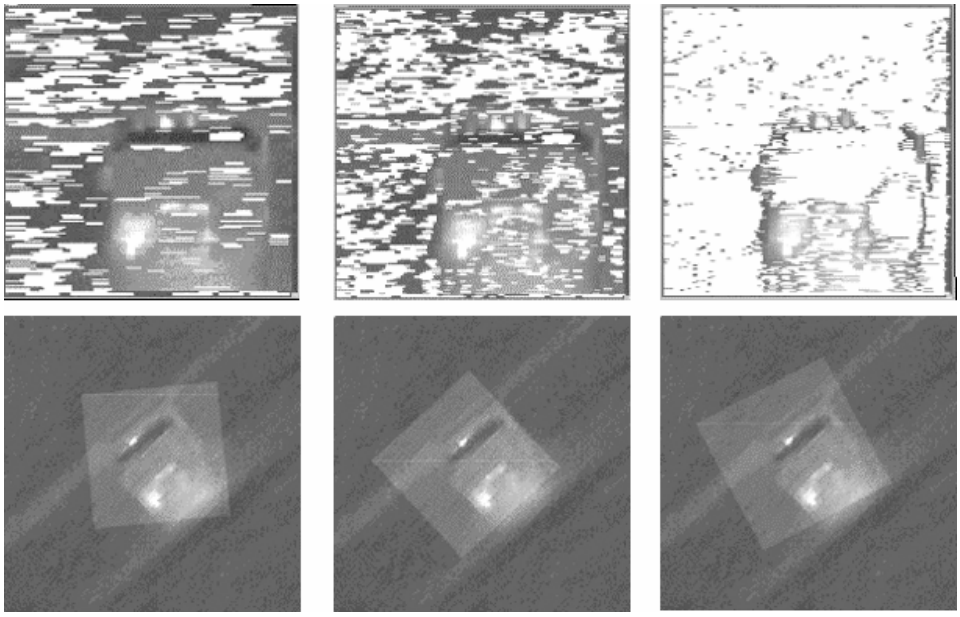

Figure 4.30: Image segmentation (top) and mapping results (bottom) for the truck image

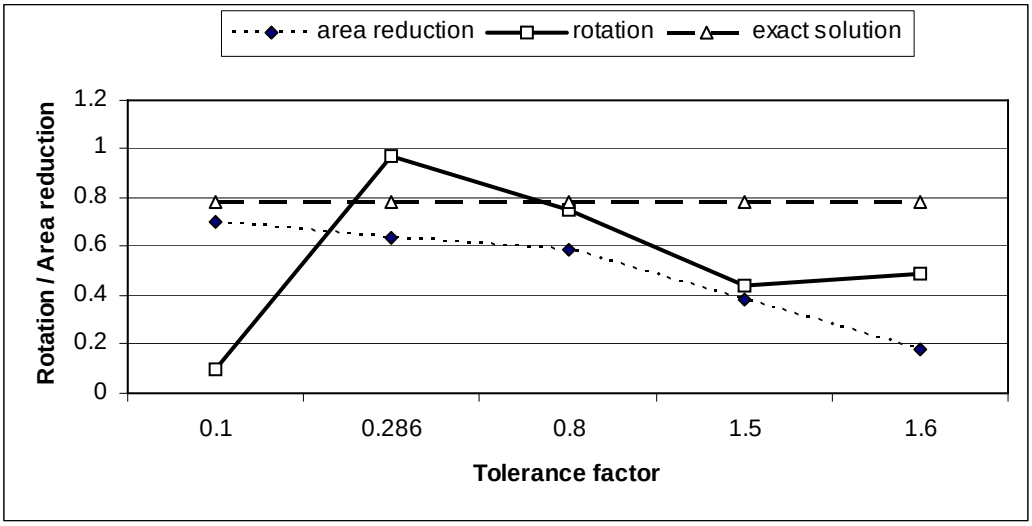

Figure 4.31: Area reduction factor Φ_A and rotation angle θ, as functions of the tolerance factor, for the segmentation of the truck image

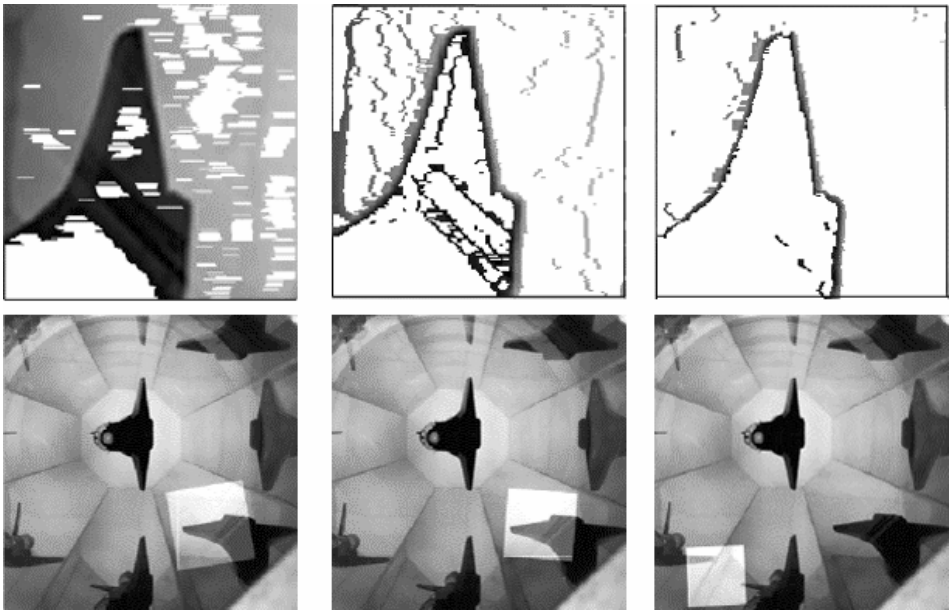

Figure 4.32: Image segmentation (top) and mapping results (bottom) for the wing image

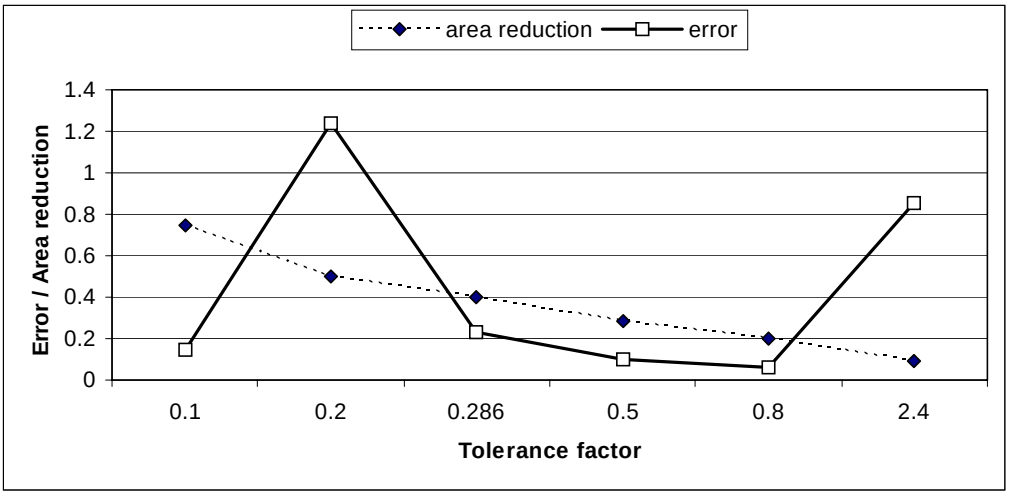

Figure 4.33: Area reduction factor Φ_A and relative error δ, as functions of the feature tolerance, for the segmentation of the wing image

Results for the image of a wing are shown in Figures 4.32 and 4.33. In Figure 4.32, the values of tolerance factor 0.1, $\Phi_A = 0.743$, $\delta = 0.147$ (left column); tolerance factor 0.8,

Φ_A = 0.199, δ = 0.062 (central column); tolerance factor 2.4, Φ_A = 0.092, δ = 0.855 (right column). Reasonably good solutions are obtained at two points corresponding to the values of the tolerance factor 0.5 and 0.8 – see Figure 4.33. The first solution, with the vector of parameters $\boldsymbol{V}$ = {168, 243, 1.5852, 3.747} has the value of the area reduction factor Φ_A = 0.288, and the relative error δ = 0.0969. The second solution, with the vector of parameters $\boldsymbol{V}$ = {170, 236, 1.5031, 4.305} has the value of Φ_A= 0.199, and the value of δ = 0.0622.

The comparative results of the two algorithms aimed at image reduction are summarized in Table 4.7. As one can see, image reduction can significantly improve the computational performance of Evolutionary algorithms in imaging optimization. The improvement is achieved by reducing the number of pixels participating in the operations of transformation and comparison during the evaluation of fitness function. Both techniques of image reduction, fractal encoding and image segmentation perform well. The effective number of fitness evaluations after applying fractal encoding to the image of a truck decreases from 813 to 297. The segmentation technique is not as successful as fractal encoding for this image, leaving the effective number of evaluations practically on the same level (811 vs. 813). For the image of a wing, the segmentation technique gives slightly better results than fractal encoding, reducing the effective number of fitness evaluations from 1406 for the full image, to only 302, for the reduced image. Fractal encoding reduces the effective number of evaluations from 1406 to 326.

	Truck			Wing		
	Full image	**Fractal encoding**	**Segment.**	**Full image**	**Fractal encoding**	**Segment.**
DX	23	22	37	170	168	170
DY	135	137	139	239	243	236
θ	0.7250	0.7633	0.7500	1.5303	1.5559	1.5031
S	1.964	1.952	2.228	3.925	3.716	4.305
Iterations	5	5	6	8	13	8
Φ_A	1.0	0.322	0.585	1.0	0.091	0.1985
Evaluations	813	921	1386	1406	3579	1520
Effective evaluations	813	297	811	1406	326	302

Table 4.7: Comparative results for selective image reduction with fractal encoding and image segmentation

4.6.4. Conclusions on Selective Fitness Evaluation

The run time of Evolutionary algorithms in imaging optimization significantly depends on the time required for evaluating the fitness value of a chromosome during one iteration. This time can be significantly reduced if some pre-processing is employed, prior to the evolutionary search. The pre-processing procedure can eliminate the areas that do not contain important information about the image and can be omitted during the operations of transformation and comparison. The procedure does so by reducing the number of pixels participating in fitness evaluation.

Two algorithms are compared, that can be potentially used for image pre-processing: the fractal encoding, and a simple segmentation technique. Numerical experiments show that both algorithms perform well, and can be successfully applied to selective fitness evaluation. For the test problems, the fractal encoding reduces the effective number of fitness evaluations from 813 (for the full image of a truck), to only 297 for the reduced image. Similarly, the segmentation technique reduces the effective number of evaluations from 1406 (for the full image of an aircraft wing) to 302.

In general, there is no particular advantage of using more complicated and resource-consuming algorithms like fractal encoding, while a simple segmentation technique can work as well. The results of the solution of optimization problem obtained with the selectively reduced image mainly depend on the optimal and representative distribution of pixels participating in fitness evaluation, rather than on the particular algorithm used for the selective image reduction.

4.7. Summary

This Chapter analyzes the following main directions of improving the EAs performance in imaging optimization.

Incorporating efficient local search and optimization techniques into global evolutionary search can increase the convergence rate and the quality of the final solution. The hybrid EA model with the commonly used steepest descent methods is compared with the classical EA model, on a sample image mapping problem in the 3-dimensional parameter

space. The classical model obtains the minimum fitness value $F = 0.0620$ and the weighted Euclidean distance from the exact solution $\delta = 0.086$, after 54 generations. Incorporating the deterministic descent method does not show a noticeable improvement over the classical EA model, providing the minimum fitness value $F = 0.0610$, and the distance $\delta = 0.114$. Incorporating the stochastic descent method provides better solution $F = 0.0380$, with the distance $\delta = 0.022$, after 36 iterations.

Two advanced local techniques, Conjugate Gradient and Downhill Simplex methods are evaluated, as potential candidates for using in the hybrid EA model, on a sample 4-dimensional image mapping problem for two image sets. One image set has convex fitness function; the other has non-convex fitness function. Both methods perform fairly well on the convex function, with DSM having a superior performance. DSM provides minimum fitness values $F = 0.00391$ and $F = 0.00338$, with the total number of evaluations $E = 46$, and $E = 148$, respectively, for the different starting points. CGM provides minimum fitness values varying from $F = 0.00398$ to $F = 0.00812$, with the total number of evaluations varying from $E = 152$ to $E = 333$. The quality of the final solution obtained with CGM significantly degrades, as the distance from the starting point to the optimum increases, while the quality of the DSM solution only slightly decreases.

For the non-convex fitness function, the quality of the CGM solution quickly deteriorates with the increasing initial distance from the optimum. The method provides the minimum value of $F = 0.01604$, and the number of evaluations $E = 91$, at the 3% distance; it breaks down beyond the 3% distance mark. The DSM performance remains robust up to the

10% distance, where the minimum value of $F = 0.00308$, and the number of evaluations is $E = 139$.

Based on the experimental results with local search, as well as on the analysis of the problems that arise when local search is incorporated into the global EAs framework, a novel two-phase cyclic algorithm alternating random search and Downhill Simplex Method is proposed. The algorithm has the following distinctive features:

- using direct DSM search, as opposed to gradient-based methods of local search frequently used in hybrid EA models, which allows to reduce the total number of fitness evaluations and increase the robustness of the algorithm, in the case of possible irregularities of fitness function;
- adding randomness to the deterministic and algorithmically well defined DSM, which allows to relocate the simplex in the local neighborhood in an advantageous way, thus preventing the search from falling into a sub-optimum solution, and reducing the total number of fitness evaluations;
- engaging partial, rather than complete local search, in a cyclic manner, which allows to reduce the number of wasted local evaluations, in the case of discarding the local neighborhood by the global evolutionary search;
- building a tree structure, in order to keep track of the visited points in the local neighborhood, which prevents the search from re-visiting the discarded points.

Experiments on a sample image set, in the case of a sample 4-dimensional mapping, show a superior performance of the proposed algorithm, in comparison with the regular

DSM method. The modified version is able to find the minimum fitness value F = 0.00216 after 4 cycles and 34 evaluations, while the regular DSM version can find F = 0.00253 after 8 cycles and 50 function evaluations.

Comparison of the hybrid EA model with the proposed two-phase local search, with the simple parallel implementation of Simulated annealing, and Multi-start algorithm that uses DSM, shows a superior performance of the proposed HEA model on a sample image set, in the case of the 5-dimensional mapping. The HEA model gives the minimum fitness value F = 0.00220 after E = 5148 fitness evaluations, with the success rate S = 1.0. Using the same initial population, Simulated annealing can find minimum fitness values varying from F = 0.00298 to F = 0.00568, with the number of fitness evaluations varying between E = 8676 and E = 70829, and the success rate varying between S = 0.271 and S = 0.451. Multi-start algorithm fails to find the global solution using the same initial population.

Increasing mutation helps maintain the diversity of the population and prevent the search from overlooking potentially optimum solutions. The size of the mutation pool can be controlled using the proposed weighted error factor F_{wer} defined by the rate at which the gradient of the fitness function is decreasing over the number of global iterations. As the fitness of the population is flattening, the value of the error factor and, consequently, the size of the mutation pool are growing, thus introducing a larger number of new candidate solutions. Experiments on a sample test set, in the case of the 3-dimensional mapping, show that the elitist EA with the adaptive mutation pool is able to find the minimum

fitness value $F = 0.06028$, with the weighted Euclidean distance from the exact solution $\delta = 0.056$, after 18 generations. For comparison, the classical model obtains the minimum fitness value $F = 0.0620$, with the distance $\delta = 0.086$, after 54 generations. Experiments also show that the error factor can be used as one of the terminating criteria of the search. When the factor F_{wer} begins to grow exponentially, the search can be terminated, because the global minimum solution has been found.

The classical operation of mutation relies on random selection of new candidate solutions from the entire search space. In reality, however, some of the search areas are better represented in the population than the others, particularly when elitism, high-rate mutation, and local search are utilized. The proposed mechanism of mutation with memory keeps track of the usage of the entire search space during all evolutionary operators, and increases the fair usage of the space. Mutation with memory draws new individuals from the areas that have been less frequently visited and are under-represented in the chromosome population. The mechanism, therefore, helps prevent EA from pre-maturely converging to a sub-optimum solution.

Comparative experiments on two image sets undergoing a sample 4-dimensional mapping show a superior performance of the hybrid EA model that uses the proposed two-phase local search and mutation with memory, over the classical elitist EA model. The classical EA model finds the minimum fitness values $F = 0.00464$ and $F = 0.00620$, after 391 and 77 generations, with the number of fitness evaluations $E = 44213$ and $E = 8731$, for the first and second sets, respectively. The modified hybrid EA model is able to

find the minimum fitness values $F = 0.00370$ and $F = 0.00246$, after 11 and 17 generations, with the total number of fitness evaluations $E = 3421$ and $E = 5467$, for the first and second sets, respectively.

Incorporating problem-specific knowledge into the algorithm helps extract the essential information about the imaging problem, and reduce the amount of information processed during a single fitness evaluation. Two techniques for selective fitness evaluation are demonstrated and compared, fractal encoding and simple image segmentation. Comparative experiments on two image sets, in the case of a sample 4-dimensional mapping problem, show that selective evaluation based on fractal encoding reduces the effective number of fitness evaluations from $E = 813$ to $E = 297$, and from $E = 1406$ to $E = 326$, for the first and second image sets, respectively. Image segmentation does not affect the number of evaluations for the first image set (813 vs. 813); it reduces the effective number of fitness evaluations from $E = 1406$ to $E = 302$, for the second image set. Successful solution of the mapping problem obtained with the selectively reduced image mainly depends on the optimal and representative distribution of pixels participating in the fitness evaluation, rather than on the particular algorithm used for selective image reduction.

Chapter 5: Improving the Performance of Evolutionary Algorithms with Image Local Response

5.1. Introduction

One of the strong points of Evolutionary algorithms is their ability to deal with complex real-world problems, without any a priori provided problem-specific knowledge. However, the generality of EAs can make them less efficient, in comparison with some ad hoc methods that extensively rely on the problem-specific knowledge, when the latter is available. Luckily, the flexibility of EAs allows for the problem specifics to be incorporated into the general evolutionary framework. There are two principal ways of such incorporation:

- providing the algorithm with the specific data about the problem;
- providing the algorithm with a fairly general method that can autonomously obtain the needed information, and seamlessly integrate it into the evolutionary search.

Since the first approach leads to the apparent loss of the algorithm's generality, and in most cases requires human intervention, the second approach seems more desirable. Such an approach is proposed in this Chapter; it is rooted in the general ideas of sensitivity analysis, and implemented in the form of Image local response.

Throughout this Chapter, six sets of 2-dimensional grayscale images are tested, in order to validate the proposed model of Image local response: an aircraft wing [70], a boat [71],

a ship [73], a palm tree [76], a rock [77], and a truck. Each set includes the 256×256-pixel reference image Img_0 of a larger scene, and a smaller template image Img_1 of an object obtained either by cropping a section from the scene, or by taking a different snapshot – see Figure 5.1.

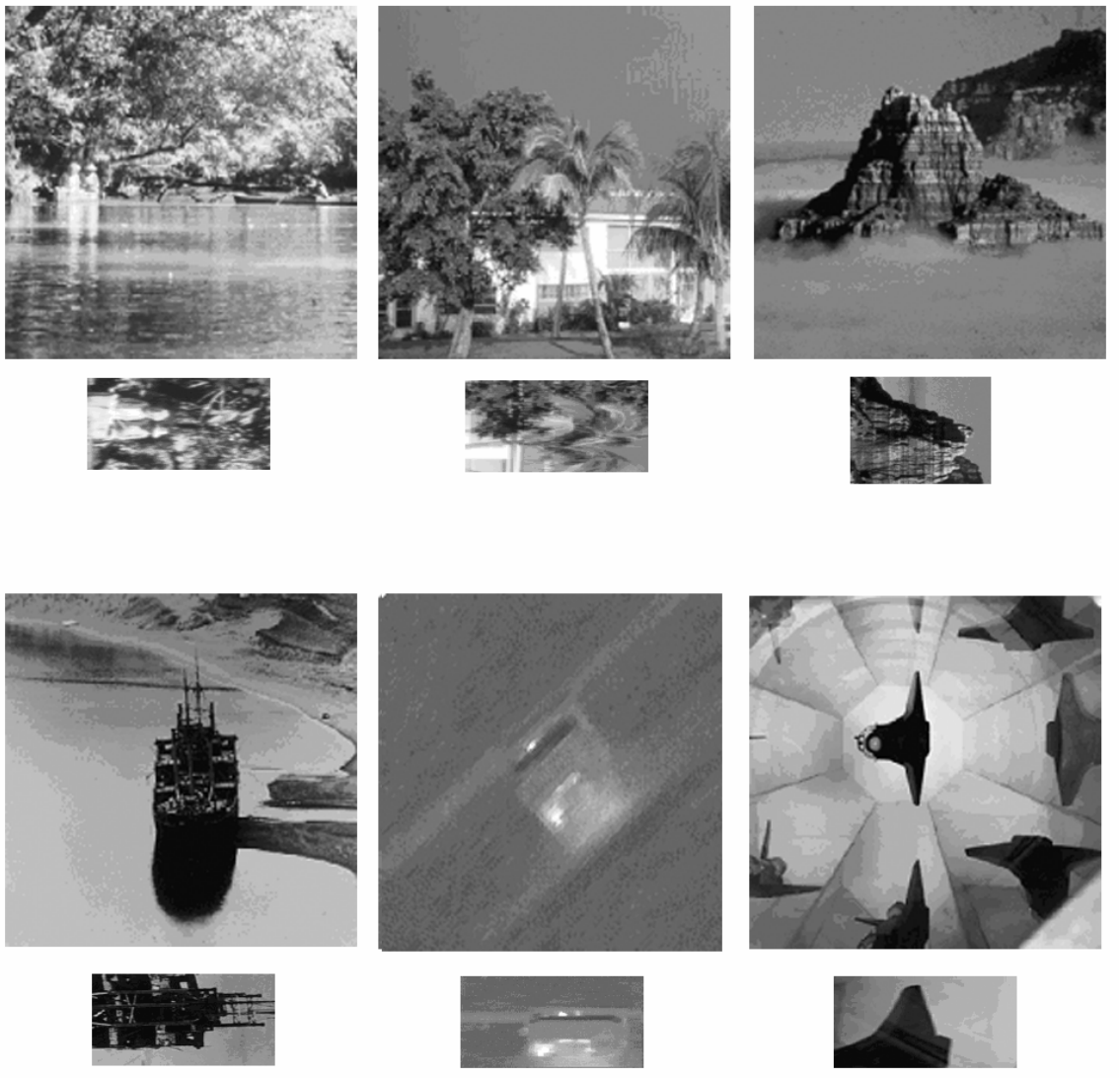

Figure 5.1: Test sets used in Chapter 5: a boat, a palm tree, a rock (top row, left to right); a ship, a truck, an aircraft wing (bottom row, left to right)

Image	Optimum parameter values				
	DX	***DY***	***θ***	***SX***	***SY***
Boat	26.0	141.0	1.571	4.0	2.0
Palm	80.0	180.0	1.571	2.53	1.27
Rock	40.0	160.0	1.571	2.0	1.5
Ship	108.0	144.0	1.571	3.56	1.78
Truck*	18.0	117.0	0.761	2.04	1.02
Wing	150.0	212.0	1.571	4.14	2.07

Table 5.1: Optimum parameter values for test images (* denotes approximate values)

The image Img_1 of each object undergoes the affine transformation defined by the 5-dimensional vector $\boldsymbol{V} = \{DX, DY, \theta, SX, SY\}$. The original problem of finding a vector $\boldsymbol{V}^*$ providing the correct mapping between the images Img_0 and Img_1 is re-formulated as the optimization problem of finding the optimum vector $\boldsymbol{V}^*$ minimizing the difference F between the images, where F is defined as the squared difference of the gray values of the images, over the area of their overlap. Since the images of a boat, a palm tree, a rock, a ship, and an aircraft wing are obtained by cropping a section from the original scene, the exact optimum values of their parameters DX, DY, θ, SX, and SY are known; they are presented in Table 5.1. The image of a truck is taken from a slightly different viewpoint, so the corresponding parameters in Table 5.1 have their approximate optimum values.

The structure of this Chapter is as follows. Section 5.2 provides the foundations of Image local response. Section 5.3 discusses experimental results related to the fundamental

properties of the response. Sections 5.4 through 5.8 give algorithmic and experimental support for using Image local response for increasing the efficiency of fitness evaluation; reducing the cost of local search, selection, and recombination; performing efficient multi-resolution and multi-sensor image analysis. Section 5.9 concludes the Chapter with the summary of the findings.

5.2. Definition and Model of Image Local Response

The concept of Image local response is somewhat related to image neighborhood and block operations [78, 121, 145], and to the node and edge functions proposed in [113]. Image local response has a fairly simple underlying idea: since the statement of the optimization problem in image mapping looks for a parameter vector $\boldsymbol{V}$ defining the unknown transformation A between the images, it seems logical to explore the response of the image to this particular type of transformation. This task can be accomplished by mapping the transformed image $Img´$ onto its original version Img, with a small transformation vector $\boldsymbol{V_u}$. Hence, Image local response R_P at a point P is defined here as the value of the difference F between the transformed $Img´$, and original Img versions of the same image, where the transformation $A_{\boldsymbol{u}}$ at the point P is small, i.e., the components of the parameter vector $\boldsymbol{V_u}$ have sufficiently small unit values. In this sense, computing Image local response is similar to computing Green's functions extensively used in mathematical physics and engineering [9]. It can be shown that there is no need to compute response R_P over the entire image, since R_P rapidly decreases, as the distance from the point P increases. It suffices to compute response R_P over a small pixel area ω_P near P, chosen as the response area.

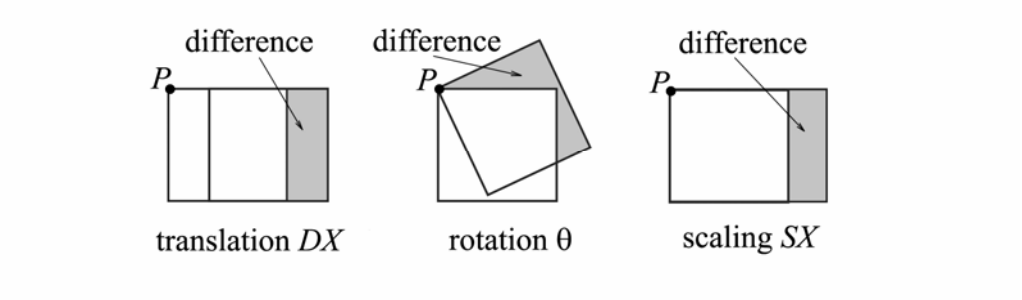

Figure 5.2: Sample local responses at point *P*, due to unit variations of the components *DX*, θ, and *SX* (from left to right)

Thus, Image local response is functionally defined here as the variation of the objective (i.e., fitness) function *F* occurred because of a small variation of the parameter vector ***V***, in the *N*-dimensional parameter space. Response is computed over a small pixel area, called the response area ω_R. For convenience and without loss of generality, a square with a side *r*, called the radius of ω_R, is chosen as the response area. For example, for affine transformation defined by five parameters: translations *DX* and *DY* along the *x* and *y* axes; rotation θ; and non-isotropic scaling factors *SX* and *SY* along the *x* and *y* axes, Image local response is evaluated at a point *P*, called the base point, as follows.

Step 1: A series of *N* partial geometric (here, affine) transformations are applied to the response area ω_R, near the base point *P* - see Figure 5.2. The shaded areas in Figure 5.2 correspond to the difference between the initial ω_R, and transformed ω_R' response areas. Each partial transformation corresponds to a unit variation of one of the *N* parameters.

Step 2: For each partial transformation, the variation of the partial difference F_i (where $i = 1,\ldots, N$) between the pixel values of the initial ω_R, and transformed ω_R' areas is computed as

$$F = \frac{\sum_{\Omega} (g_1(x', y') - g_0(x, y))^2}{\Omega^2}, \tag{5.1}$$

where $g_1(x',y')$ and $g_0(x, y)$ are the gray values of the images Img_1 and Img_0, respectively, and $\Omega = \omega_R$.

Step 3: Response R_P at the point P is computed as the averaged sum of all N differences F_i ($i = 1,\ldots, N$), as follows:

$$R_P = \frac{\sum_{i=1}^{N} F_i}{N}. \tag{5.2}$$

From the algorithmic point of view, the operation of computing local response is similar to the operation of image filtering [78]. Very much like filtering, computing local response also produces a new output image Img_R, which serves as the graphical representation of the image response matrix M_R. Like filtering, computing local response is also a neighborhood operation, in which the value of any given pixel P in the output image Img_R is determined by the means of applying the described procedure to the pixel values in the local neighborhood ω_P of the pixel P in the input image (i.e., in the original image Img).

The important property of the response is the rate at which it changes when the radius r of the response area increases. It would be tedious and unnecessary to calculate the exact rate, because pixel distribution $g(x,y)$ is a stochastic process; it varies not only for different images, but for different points within the same image, as well. However, a simple uniform distribution can give an idea of the principal trend in the response change. Let us assume that $g(x,y) = a$ inside the initial response area ω_R, and $g(x,y) = b$ outside the area, where both a and b are constants. Then the gray values of the initial and transformed response areas will differ only within the shaded sub-areas shown in Figure 5.2. As the radius r increases, the sub-areas and, consequently, the difference of the gray values are growing at a different rate, e.g., proportionally to r for translations, and proportionally to r^2 for rotation. When the partial difference F_i is computed, the difference of the gray values is divided by the squared value of the response area. It is easy to see now that the rate at which the partial differences F_i change with the increasing radius r will have the following tight bounds:

- for translations DX and DY, $F_1 = O(\gamma_1/r^3)$ and $F_2 = O(\gamma_2/r^3)$, respectively;
- for rotation θ, $F_3 = O(\gamma_3/r^2)$;
- for scaling factors SX and SY, $F_4 = O(\gamma_4/r^2)$ and $F_5 = O(\gamma_5/r^2)$, respectively.

In general, the unknown factors γ_i (i = 1,…, N) are the random functions of the particular pixel distribution. According to the averaging formula (5.2), the tight bound for the response value R is dominated by the term r^{-2}, and is equal to $O(\gamma/r^2)$, where γ is the averaged function of the pixel distribution. In the derivations below, the following two simplified assumptions will be used:

1. The unknown averaged function γ of the actual pixel distribution can be represented by a single constant value, e.g., by its mean value μ. The assumption is intuitively justified by the integral effect of the squared pixel differences, when the local response is computed. The assumption will only simplify the derivation of the final formula; it will not affect the outcome, since the response values are eventually computed for the actual pixel distribution.
2. The tight bound $R = O(\gamma/r^2)$ is relaxed, and replaced with a more conservative and cautious estimate $R = O(\gamma/r)$. This assumption ensures that even with the random function γ, the response R still decreases at a high rate of r^{-1}.

Under the above assumptions, Image local response has some important and convenient properties that will be extensively used throughout this Chapter:

1. As the radius r of the response area or, alternatively, the distance l from the base point P increases, the response R_P rapidly decreases, as it is inversely proportional to r.
2. For any two points P and Q having similar pixel distributions and situated relatively close to each other, the difference between their respective responses R_P and R_Q is fairly small.
3. If the pixel distribution in the response area has a steep change (e.g., near the object edges in the image), the response R will have the corresponding increase in its value, near the point or area of the change.

The first property assures that the value of the local response R_P computed over a small area of the image can be used as a fairly good approximation to the response R_{Img} computed over the entire image. But the value of R_{Img}, in turn, corresponds to the value of the fitness function F computed for a small transformation $A_{\boldsymbol{u}}$ of the image *Img*. Since *Img* itself constitutes the optimum solution to the problem of mapping the transformed version *Img´* onto its original version, under the small transformation $A_{\boldsymbol{u}}$, the local response R_P can be used as a fairly good approximation to the fitness function F computed at a point P, in the vicinity of the optimum solution.

The second and the third properties indicate that the values of local response reflect the degree of smoothness of the fitness function F in the vicinity of the optimum solution. When F is fairly smooth, the distribution of the response values is relatively flat. On the contrary, any noticeable variation of F will cause the corresponding significant variation of the response values.

Once Image local response has been defined, the following inverse problem can be stated. Given local response R_P, one can look at the “deformation” of a small local area near P caused by R_P. If R_P is sufficiently small, the deformation of the area is small. If the value of the response increases, the locality undergoes the corresponding deformation of contraction, i.e., it shrinks. A simple one-dimensional model can be utilized to derive the estimate of the deformation of the locality near the base point P caused by the response R_P.

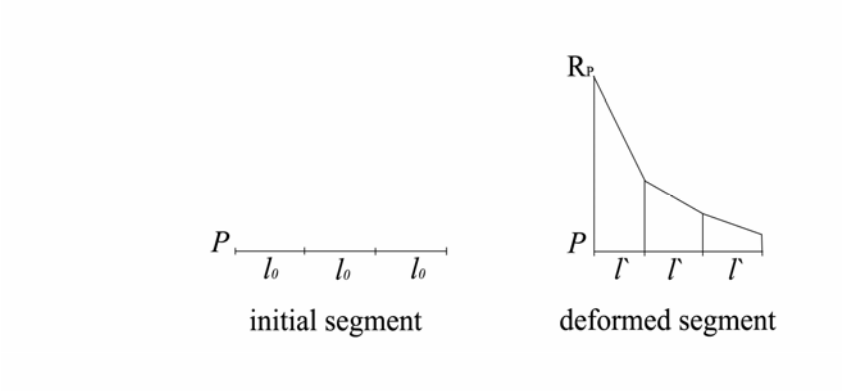

Figure 5.3: Longitudinal deformation of contraction of a line segment caused by the response R_P

Figure 5.3 shows the deformation occurring in a small line segment L near the base point P, as the response value R increases. When R has its minimum value, each elementary length l of the segment equals its initial value l_0. As R grows, the segment continues to shrink, and its elementary length becomes $l' < l_0$. Since the degree to which the segment shrinks is proportional to the value of the response, the relative deformation, or the longitudinal "strain" of the element l can be written as

$$\frac{l_0 - l'}{l_0} = \frac{\Delta l}{l_0} = cR\,, \tag{5.3}$$

where c is a constant, in accordance with the first simplified assumption. Since only the relative deformation between different points of an image is of interest, the expression for the elementary strain can be normalized, and the value of the constant c can be set to 1.

Asymptotically, for a sequence of $M \to \infty$ elements of the segment L, each of the initial length $l_0 \to 0$, summing up the left side of Formula (5.3) yields the following longitudinal strain ε of the entire segment:

$$\sum_{M}\left(\frac{l_0 - l'}{l_0}\right) \Rightarrow \frac{L_0 - L'}{L_0} = \frac{\Delta L}{L_0} = \varepsilon\,, \tag{5.4}$$

where L_0 and L' are the initial and deformed lengths of the segment, respectively.

In accordance with the second simplified assumption stating that R is inversely proportional to the distance l from the base point, summing up the right side of Formula (5.3) asymptotically yields

$$\sum_{M}(R) \Rightarrow \int_{L'}(R_P / l)dl = (\ln R_P - \ln R_{P\min})\,, \tag{5.5}$$

where $R_{P\min}$ is some (small) value of R at the distance L' from the base point P. Combining Formulae (5.4) and (5.5) yields the following expression for the longitudinal strain of the line segment L near the point P:

$$\varepsilon = \frac{\Delta L}{L_0} = \ln(R_P / R_{P\min}) = \ln(R_P L')\,. \tag{5.6}$$

If the line segment L_{PQ} is enclosed between two points P and Q having their respective response values R_P and R_Q, it will experience the integral effect from the both response points. The strain ε of the segment can be computed then as

$$\varepsilon = \frac{\Delta L}{L_0} = \frac{L_0 - L'}{L_0} = \ln(R_P L') - \ln(R_Q L') = \ln(R_P / R_Q)\,, \tag{5.7}$$

where L_0 and L' are the initial and deformed lengths of the segment L_{PQ}, respectively.

The length L' of the deformed segment L_{PQ} can be derived now from Formula (5.7) as

$$L' = L_0\,(1 - \varepsilon) = L_0\,(1 - \ln(R_P / R_Q))\,. \tag{5.8}$$

Since the line segment L_{PQ} experiences the deformation of contraction, the following condition should hold:

$$0 < (1 - \ln(R_P / R_Q)) \leq 1. \quad (5.9)$$

Formula (5.8), together with the condition (5.9), serves as an approximate model used to derive the adaptive control mechanism for the step size in the modified version of Downhill Simplex Method.

5.3. Computational Experiments with Image Local Response

A set of experiments is performed on the image of a ship, which has a purpose of looking at the behavior of Image local response as a function of the pixel distribution. Response is computed for every point of the image, and the actual properties of the response are analyzed and compared with the desired properties stated in section 5.2 of this Chapter. Figure 5.4 shows the image of a ship [73], with the indicated five characteristic points labeled 0 through 4. Point 0, with the coordinates $\boldsymbol{X} = \{20,20\}$, has a locality with a smooth, non-homogeneous pixel distribution, whose gray values range from 45 to 144. Points 1 and 2, with the coordinates $\boldsymbol{X} = \{50,150\}$ and $\boldsymbol{X} = \{70,170\}$, respectively, are located in a relatively flat distribution area, with the gray values ranging between 173 and 180, near the point 1, and between 174 and 182, near the point 2. Points 3 and 4 have their respective coordinates $\boldsymbol{X} = \{90,190\}$ and $\boldsymbol{X} = \{100,200\}$, and are situated close to the object edge, where the abrupt change of the gray values occurs.

Figure 5.4: Image of a ship with the identified response points 0 – 4

Figure 5.5 shows the actual response values at the characteristic points, as functions of the radius of the response area. Points 0 and 1 have smooth pixel distributions, and their responses are smooth monotonic functions of the radius r. Both functions have $O(1/r^2)$ and $O(1/r)$ as the lower and upper bounds, respectively. Point 1 is situated in the homogeneous locality with the narrow range of gray values, and is closer to its lower bound of $O(1/r^2)$. Point 0 has a non-homogeneous pixel distribution with a broad range of gray values; its response deviates from the $O(1/r^2)$ bound, and comes closer toward the upper bound of $O(1/r)$. The response at point 0 also has significantly higher values than the response at point 1. One can conclude that local response tends to follow the upper bound $O(1/r)$, as the non-homogeneity of the locality grows. This behavior provides the experimental support for the validity of the first property stated in section 5.2.

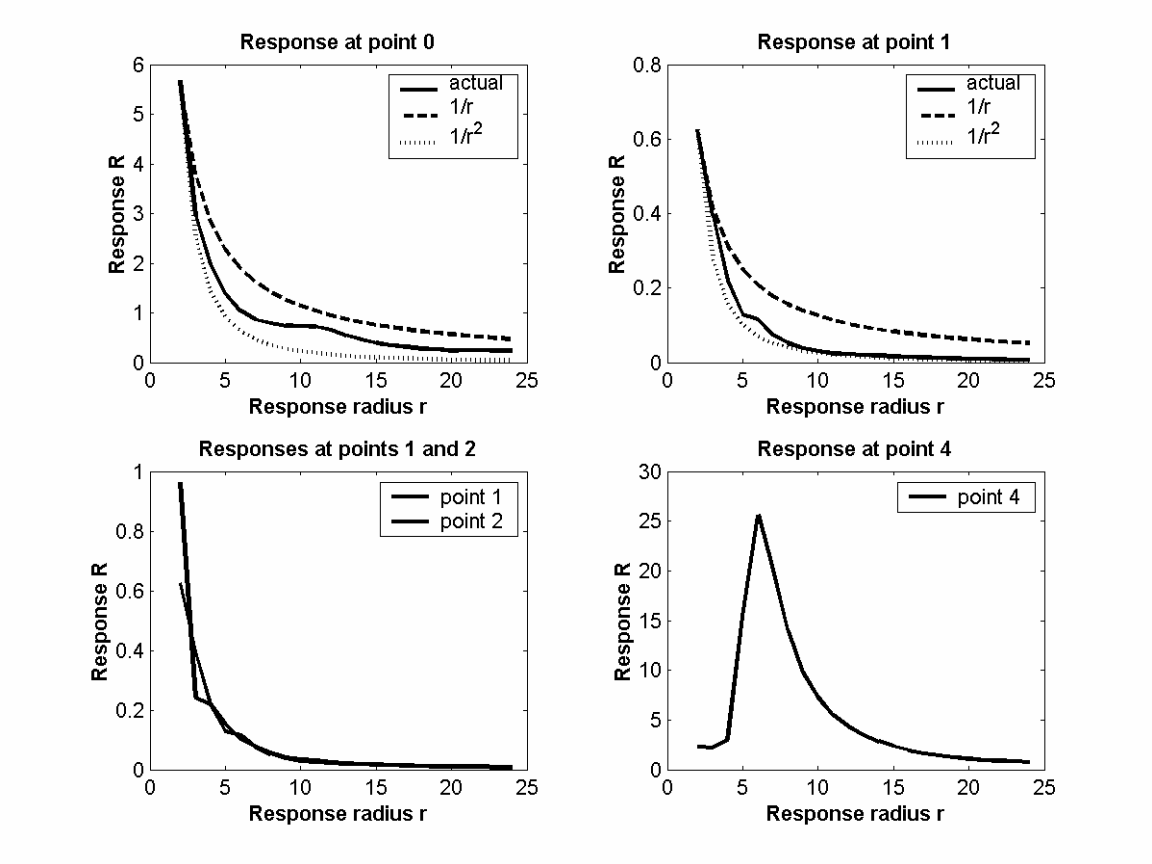

Figure 5.5: Behavior of Image local response at the characteristic points 0, 1, 2, and 4 of the ship image

Points 1 and 2 are situated close to each other in the area of a relatively flat pixel distribution. As can be seen from Figure 5.5, response functions at both points differ for the initial radius value $r = 2$, and gradually merge near the value $r = 7$. The differences at small radius values are due to the strong effect of individual pixels when the response area is small. As the radius increases, the response area grows, and the averaging of the gray values over the response area outweighs the effect of individual pixels, thus making the two response functions almost identical. This behavior provides an experimental support for the second property of local response stated in section 5.2. The minimum

radius r_{min} at which two points closely situated in the area with a relatively flat pixel distribution have nearly identical response values can be recommended as the optimum response radius during the execution of Evolutionary algorithms. The value r_{min} provides the minimum cost of computing local responses while ensuring their sufficient averaging degree. Figure 5.5 clearly shows the hike in the response value at point 4, where the response area crosses the edge of the ship shadow marked by the abrupt change of the corresponding gray values. Point 3 also experiences the hike, which is much lower than the hike at point 4, and is offset at the corresponding distance from point 4. This behavior supports the third response property stated in section 5.2.

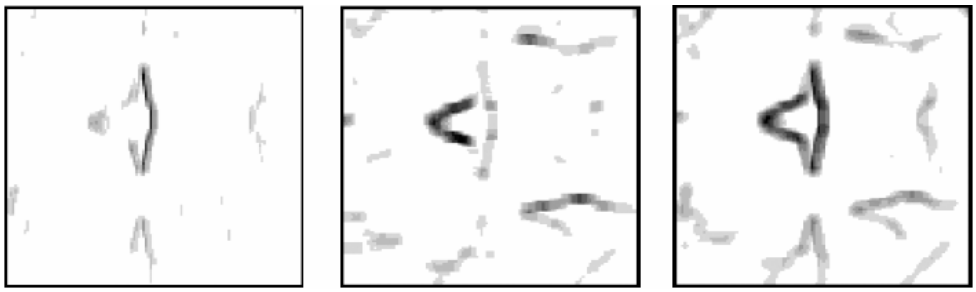

Figure 5.6: Sample inverse images of the *DX* (left), θ (center), and the average (right) responses with the response radius $r = 10$, for the wing image

Similar results are obtained for the wing image [70]. Figure 5.6 shows the sample inverse images of local responses of the wing image, due to the unit variations of the parameter vector. The radius r of the response area ω_R is set to 10 pixels. Some interesting response points in the image are indicated with the numbers 1–6 in Figure 5.7. The group of points 1–3 is located in the area where the gray value distribution is relatively flat, so we can expect a relatively smooth behavior of the corresponding response values. On the

contrary, the group of points 4–6 is located near the object edges, which should spur the increase in the response values for these points, according to the proposed response model.

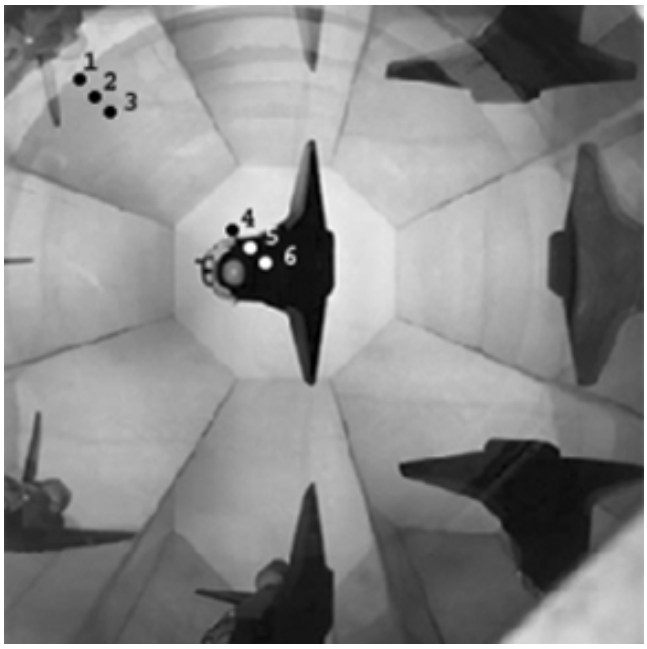

Figure 5.7: Image of a wing with the identified response points 1 – 6

The average response values of the base points 1–3 are shown in Figure 5.8. As one can see, the responses of all three points smoothly and rapidly decrease as the distance from the base point increases. Since the distribution of gray values in this area of the image is relatively flat, all responses rapidly converge to the same value at a distance $d \geq 10$. Similarly, Figure 5.9 shows the average response values for the base points 4–6. All responses experience significant increase in their values near the object edges. As the distance d from the edge increases, each response function returns to its regular shape $R = f(1/d)$. These results are consistent with the proposed micro-level model of Image local

response described in section 5.2. Therefore, the model of Image local response can be used as an indicator of local properties of the gray value distribution in the image.

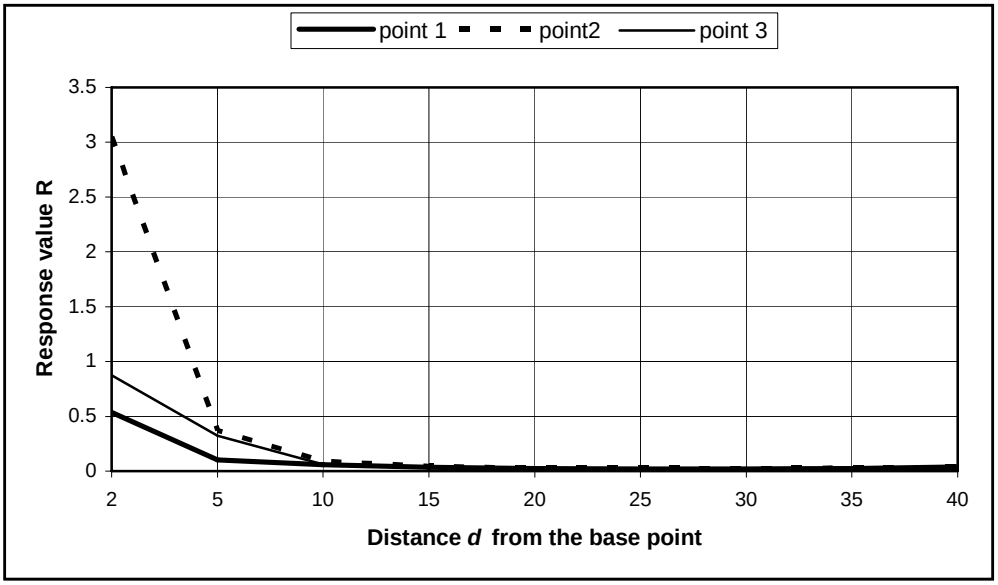

Figure 5.8: Average local responses at points 1–3 of the wing image

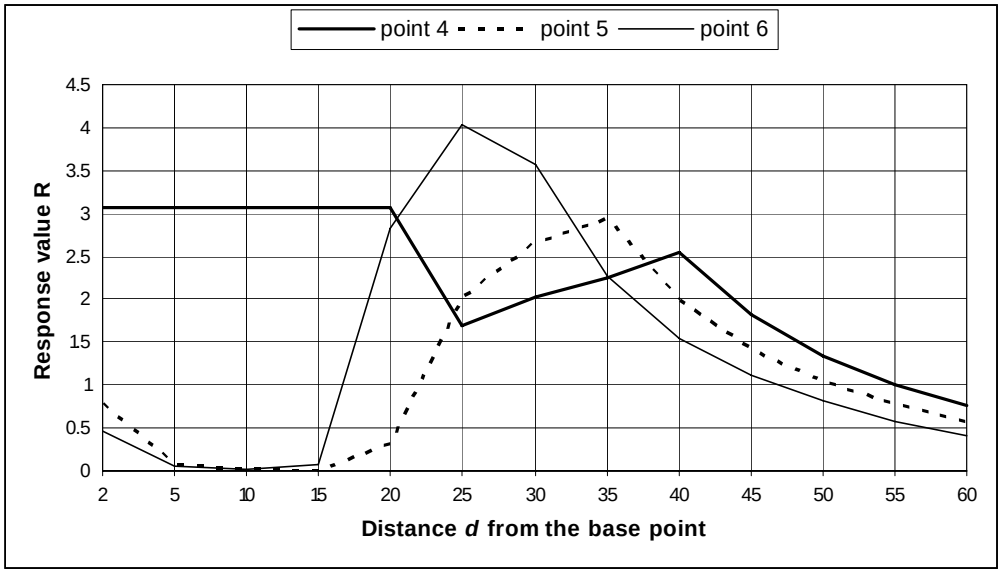

Figure 5.9: Average local response at points 4–6 of the wing image

5.4. Reduction of Computational Cost of Fitness Evaluation

The total computational cost T_J of the HEA run is comprised of the cost of fitness evaluations and the overhead related to various evolutionary and bookkeeping operations, as shown in Chapter 4 (section 4.6). Make a note that the number of fitness evaluations for a chromosome $\boldsymbol{V_g}$ equals $E_{jg} = 0$, if the chromosome $\boldsymbol{V_g}$ has been already evaluated at one of the previous iterations; $E_{jg} = 1$, if the chromosome is new in the population; and $E_{jg} = l_{jg}$, if the chromosome is in the reproduction list, where l_{jg} is the number of fitness evaluations associated with the local search around the chromosome $\boldsymbol{V_g}$, at iteration j.

In a typical real-world application of HEA, the lion's share of the computational cost T_J is attributed to fitness evaluations. This section focuses on the reduction of the time t_e required for one operation of fitness evaluation, using Image local response. One operation of fitness evaluation for a chromosome $\boldsymbol{V_g}$ includes the following operations:

- transformation of the template image Img_1;
- pixel-wise comparison of the transformed image Img_1' with the original image Img_0 of the scene;
- evaluation of the difference F_g between the images.

If the template image has $M \times N$ pixels, then each of the above operations has to be performed $M \times N$ times. Hence, the reduction of the number of pixels participating in a single fitness evaluation could result in a significant reduction of the total cost of the algorithm. The effect of the reduction can be estimated using the area reduction factor $\Phi_A < 1$ defined in section 4.6 of Chapter 4 as

$$\Phi_A = \frac{\Omega'}{N \times M}, \tag{5.10}$$

where Ω' is the reduced area corresponding to the reduced number of pixels participating in a single fitness evaluation. If C_r is the number of fitness evaluations using the reduced area Ω', then the equivalent number of evaluations C_e corresponding to the full image is defined as

$$C_e = \Phi_A \, C_r. \tag{5.11}$$

It would be desirable for the reduction method to facilitate the search while preserving the essential characteristic features of the image. Such a method can be derived based on utilizing Image local response. As follows from the functional definition of the response, the latter accentuates those segments of the image that change the most during its transformation. Therefore, the response extracts an important feature of the image, its dynamic contents. The total area Ω of the image can be now represented as the sum of the dynamic Ω_D and static Ω_S contents, as follows:

$$\Omega = \Omega_D + \Omega_S, \tag{5.12}$$

where the area Ω_D will have the most significant impact on the fitness evaluation for the transformed image Img_1. The algorithm using Image local response in image reduction and selective fitness evaluation can be formulated now as follows.

Step 1: The matrix M_R of Image local response for the image Img_1 is computed during the pre-processing stage.

Step 2: The response threshold T_R defining the dynamic contents Ω_D is chosen, such that all segments of the image with their response values below T_R are excluded from the fitness evaluation.

Step 3: The $M \times N$ bit mask of the image is formed, where the segments with their response values below T_R (i.e., the static area Ω_S) have 0s, while the segments with their response values above T_R (i.e., the dynamic area Ω_D) have 1s.

Step 4: The bit mask is used at the beginning of the evolutionary search, in order to compute fitness F only over those segments of the image that correspond to 1s in the bit mask. In the process of the evolutionary search, the procedure computing fitness values F switches to the full image evaluation, once F of the best chromosome falls below some pre-set fitness threshold T_F.

In order to validate the possibility of using Image local response to reduce the computational cost of fitness evaluation, computational experiments are conducted on the images of a ship and a wing shown in Figure 5.1, section 5.1 of this Chapter. Each set includes the 256×256-pixel reference image Img_0 of the scene, and the 256×128-pixel template image Img_1 obtained by cropping a section from the scene. The image Img_1 of each template is transformed using the 5-dimensional vector $\boldsymbol{V} = \{DX, DY, \theta, SX, SY\}$. The translations DX and DY, and the rotation θ define the location of the template in the scene (the rigid body transformation), and the non-uniform scaling factors SX and SY define the local distortion of the template. As one can see from Figure 5.1, both templates

are significantly distorted, i.e., stretched along the *x* axis, with the ratio $SX / SY = 2$. The exact optimum values of the parameters *DX*, *DY*, θ, *SX*, and *SY* for the both images are known and presented in Table 5.1, section 5.1 of this Chapter.

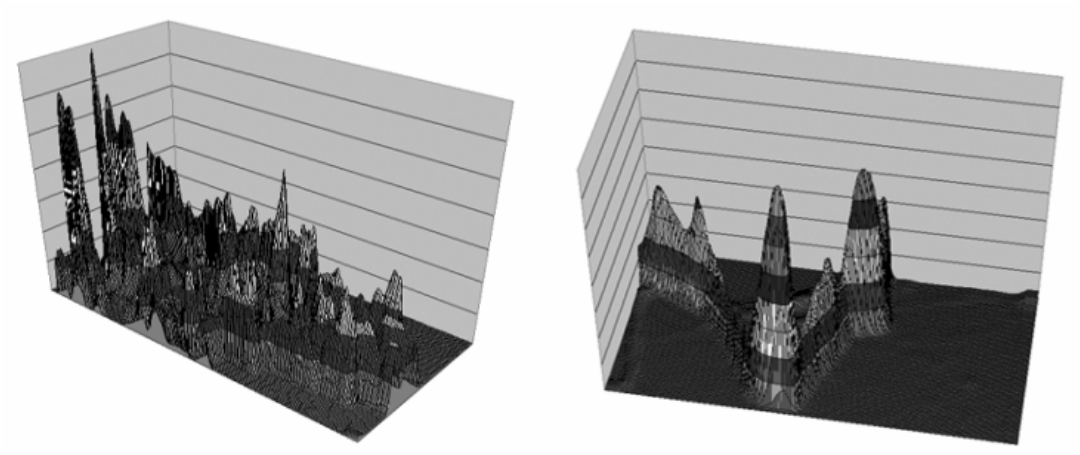

Figure 5.10: 3-D view of the template response matrix M_R: a ship (left) and a wing (right)

Figure 5.10 shows the 3-D views of the response matrices M_R computed for the template images. The thresholds $T_R = 4.0$ and $T_R = 0.5$ are applied to the response matrices of the ship and wing, respectively. Figure 5.11 gives the inverse 2-D views of the both response matrices after applying the thresholds, as well as the images of the corresponding bit masks. During the evolutionary search, only those image segments that have 1s in the corresponding bit mask participate in the fitness evaluation. The fitness threshold T_F is set to $T_F = 0.19$ and $T_F = 0.45$, for the ship and wing images, respectively. When the fitness value of the best chromosome falls below the threshold T_F, the evaluation procedure switches to the full image evaluation.

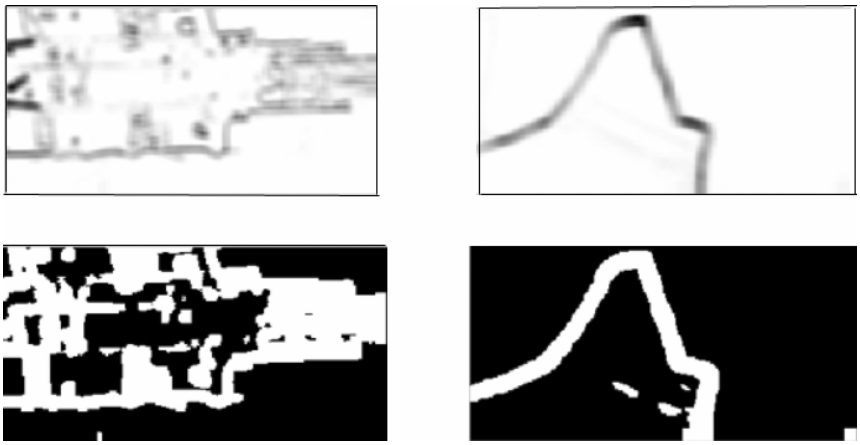

Figure 5.11: Inverse view of the response matrix M_R (top row) and the corresponding bit mask (bottom row): a ship (left) and a wing (right)

The area reduction factor for the ship image is $\Phi_A = 0.408$, and for the wing image $\Phi_A = 0.12$. The number C_r of fitness evaluations with the bit masks is $C_r = 1836$ and $C_r = 8598$, for the ship and wing images, respectively. The equivalent number of evaluations is $C_e = 749$ for the ship, and $C_e = 1032$ for the wing. The number of fitness evaluations C_f with the full image after the threshold T_F has passed, is $C_f = 5339$ and $C_f = 1463$ for the ship and wing images, respectively. The total number of evaluations $C = (C_f + C_e)$ is $C = 6088$ and $C = 2495$, for the ship and wing images, respectively.

The total number of fitness evaluations using the algorithm without image reduction is $C = 9119$ and $C = 5236$, for the ship and wing images, respectively. One can immediately see that the use of the response analysis for reducing the area that participates in fitness evaluation, results in a significant reduction of the computational cost associated with the evaluations. The total number of evaluations is reduced from 9119 to 6088 for the ship image (33% savings), and from 5236 to 2495 for the wing image (52% savings). The

optimum values of the parameter vectors are virtually the same; they are very close to their exact values. Table 5.2 summarizes the findings of the experiments, and Figure 5.12 shows the final mapping results.

Image	**Optimum parameter values with full image evaluations**				
	DX	***DY***	θ	***SX***	***SY***
Ship	108.7	146.2	1.57	3.52	1.74
Wing	149.9	213.4	1.58	3.96	2.06
	Optimum parameter values with reduced image evaluations				
	DX	***DY***	θ	***SX***	***SY***
Ship	109.1	146.9	1.57	3.51	1.79
Wing	150.3	211.4	1.58	4.27	2.15
	Attributes with full image evaluations				
	Number of generations		**Number of evaluations**		**Fitness**
Ship	20		9119		0.00876
Wing	11		5236		0.00317
	Attributes with reduced image evaluations				
	Number of generations		**Number of evaluations**		**Fitness**
Ship	17		6088		0.01154
Wing	23		2495		0.00248

Table 5.2: Optimum parameters and number of fitness evaluations

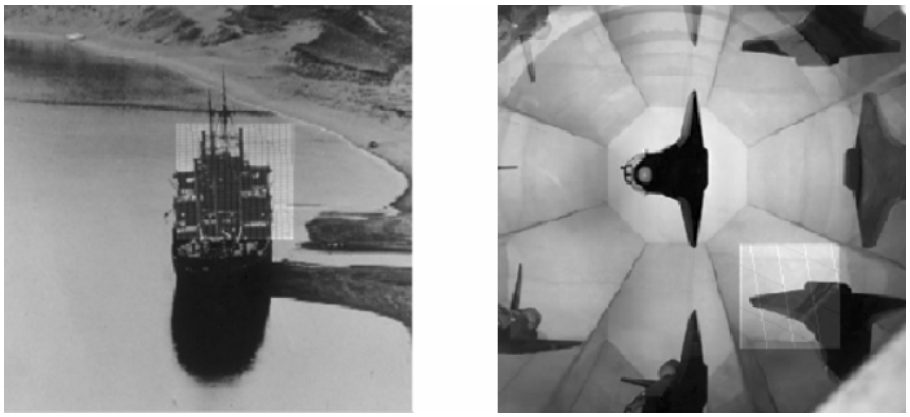

Figure 5.12: Results of image mapping with HEA and reduced fitness evaluation: a ship (left) and a wing (right)

5.5. Adaptive Control Mechanism in Local DSM Search

Local search can significantly improve the overall performance of Evolutionary algorithms, but does so at the extra cost of additional evaluations of fitness function F. In the case of using DSM, as the hybrid algorithm iterates, the total number of cycles L_C of the local DSM search is growing as

$$L_C = \sum_{i=1}^{I} \sum_{j=1}^{S_i} C_{ij} \,, \tag{5.13}$$

where I is the total number of the global EA iterations to date, S_i is the length of the list of fitter chromosomes participating in the DSM search at iteration i, and C_{ij} is the number of DSM cycles for the chromosome j, at iteration i.

The total number of fitness evaluations L_E spent on the DSM search grows as

$$L_E = \sum_{i=1}^{I} \sum_{j=1}^{S_i} \sum_{k=1}^{C_{ij}} e_{ijk} \,, \tag{5.14}$$

where e_{ijk} is the number of function evaluations for the chromosome j in the fitter chromosome list during the cycle k, at iteration i. Since the number of local fitness evaluations L_E can constitute a significant fraction of the total number of fitness evaluations during the EA search, the problem arises of reducing L_E.

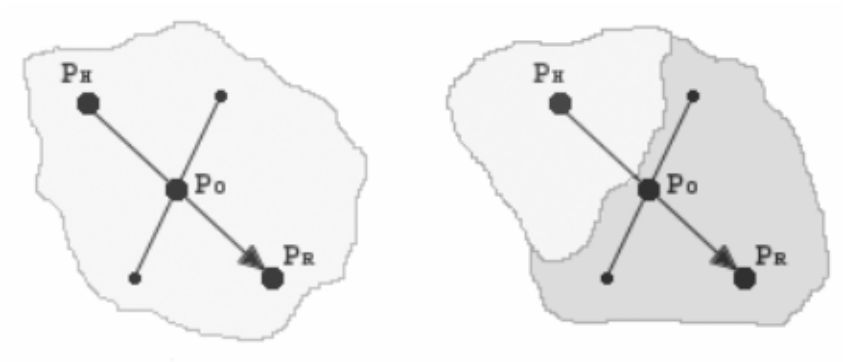

Figure 5.13: Operation of reflection in the DSM search performed on a smooth (left) and a rough (right) gray value surfaces

The drawback of the regular DSM algorithm is that the step at which the simplex moves or changes its shape does not depend on the absolute values of the objective function (i.e., fitness function in HEA) at the vertices; nor does it depend on the differences between these values. The walk of the DSM simplex is controlled only by the ranking order of its vertices, and by the values of the DSM coefficients (α_1, α_2, α_3, α_4). The coefficients have fixed values, which are usually set to (-1, 2, 0.5, 0.5) [124]. In application to image mapping, the fixed span of the simplex movement means that the simplex moves at the same speed and with the same step size on both, smooth and rough gray value surface, as long as its vertices are aligned in the same ranking order, as illustrated in Figure 5.13.

Intuitively, it would be desirable for the simplex to move at a regular speed on a relatively smooth gray value surface, but to slow down and take smaller steps on a rough surface, near the object edges, or other noticeable changes in the gray value distribution. In order to adaptively control the step size, additional points in the vicinity of the simplex have to be evaluated. But the computational cost associated with the evaluation of the objective function is exactly what one is trying to reduce. Therefore, a model has to be designed that would sense the local properties of the function landscape in the neighborhood of the DSM simplex, without the overhead of the expensive computation of the complete function values.

The properties of Image local response stated in section 5.2 of this Chapter make it an efficient technique for reducing the overall computational cost of fitness evaluations associated with the local DSM search. In order to make the simplex adaptive to the properties of the objective function *F* defined as the squared difference of the gray values of the images, over the area of their intersection, the model of Image local response is utilized to control the length of the vector $\boldsymbol{\alpha} = \{\alpha_1, \alpha_2, \alpha_3, \alpha_4\}$ of the DSM coefficients. If the surface of *F* in the locality is sufficiently smooth, its response is nearly flat, and the locality experiences only a slight deformation. Then the components of the vector $\boldsymbol{\alpha}$ take on their standard values $\boldsymbol{\alpha} = \{-1,2,0.5,0.5\}$. However, if function *F* significantly changes, the correspondingly changing response will cause the deformation of contraction of the locality, resulting in the contraction of the vector $\boldsymbol{\alpha}$. The degree of the contraction of $\boldsymbol{\alpha}$ can be estimated from Formula (5.8), derived in section 5.2 of this Chapter.

Putting everything together, a contraction transformation $T(\boldsymbol{\alpha})$ of the vector $\boldsymbol{\alpha} = \{\alpha_1, \alpha_2, \alpha_3, \alpha_4\}$ of the DSM coefficients between two points, P and Q, is defined in the following form:

$$\boldsymbol{\alpha}' = T(\boldsymbol{\alpha}) = D_{PQ}\boldsymbol{\alpha}\,, \tag{5.15}$$

where $\boldsymbol{\alpha}' = \{\alpha_1', \alpha_2', \alpha_3', \alpha_4'\}$ is the vector of the modified DSM coefficients, and D_{PQ} is the diagonal matrix of the response coefficients C_{PQ} estimated from Formula (5.8), as

$$C_{PQ} = (1 - \ln(R_P / R_Q))\,, \tag{5.16}$$

where R_P and R_Q are the response values at the points P and Q, respectively. It is worth to note that the averaging operation over all N components of the parameter vector $\boldsymbol{V}$ for the response R reduces the risk of a possible significant distortion of the simplex, after the transformation $T(\boldsymbol{\alpha})$ has been applied to the coefficient vector $\boldsymbol{\alpha}$.

The coefficient C_{PQ} is computed for every pair of points participating in the DSM simplex. It can be seen that C_{PQ} in Formula (5.16) satisfies the following important properties:

- $C_{PQ} \approx 1$ for a smooth surface of the function F, with the small variation of the response, i.e., when $R_P \approx R_Q$;
- $C_{PQ} < 1$ for a rough surface of the function F, with the large variation of the response, i.e., $R_P \neq R_Q$;
- C_{PQ} decreases, as the roughness of the surface of the function F increases.

In order to validate the utilization of Image local response for controlling the step size of the local DSM search, and to evaluate the computational efficiency of the proposed

approach, 2-D images presented in Figure 5.1 (section 5.1 of this Chapter) are tested. A series of 100 runs for each test set are performed, where for each run, the vertices of the initial DSM simplex are randomly placed in the neighborhood of the optimum solution in the following manner. The value of each component of the vector $\boldsymbol{V}$ for each of the six vertices is independently drawn from the (±10%) range of the corresponding domain centered at the component's optimum value, with the uniform probability. For example, the translation *DX* for a 256×256 - pixel image has the domain range of 0 – 255. Correspondingly, the value of *DX* for the image is drawn from the interval (40.0 ± 25.5), i.e., from the interval (14.5, 65.5), where *DX* = 40.0 corresponds to the optimum value of the component *DX*.

Image	**Value**	***DX***		***DY***		$\boldsymbol{\theta}$		***SX***	
		Reg	**Rsps**	**Reg**	**Rsps**	**Reg**	**Rsps**	**Reg**	**Rsps**
Ship	**Mean**	109.3	109.1	145.3	145.1	1.60	1.60	3.6	3.6
	St.dev	0.79	0.91	0.79	0.91	0.02	0.02	0.01	0.01
	Max	110.4	110.3	146.4	146.3	1.63	1.63	3.6	3.6
	Min	107.0	107.0	143.0	143.0	1.55	1.55	3.5	3.5
Wing	**Mean**	149.7	149.7	211.7	211.7	1.56	1.56	4.1	4.1
	St.dev	0.12	0.11	0.12	0.11	0.00	0.00	0.00	0.00
	Max	149.8	150.0	211.8	211.9	1.57	1.57	4.1	4.1
	Min	149.3	149.3	211.3	211.3	1.55	1.55	4.1	4.1

Table 5.3: Optimum parameters over 100 runs, for the regular DSM (Reg) and its response-enhanced modification (Rsps)

The series of 100 runs is performed for the regular (i.e., constant) values of the DSM coefficients, and for the modified (i.e., adaptive) DSM coefficients. The latter are computed according to Formulae (5.15) and (5.16) using Image local response as a means to adaptively control the values of the coefficients. The sample comparative results for a ship and a wing are presented in Table 5.3, in terms of the mean values, standard deviations, and maximum and minimum values of the final optimum parameters, over all 100 runs.

By and large, the regular DSM algorithm is able to find almost exact parameter values for all images, except for the truck image. The maximum relative error of 1.85% among the five images occurs for the value of the rotation angle θ, for the ship image. Since only the approximate estimates for a truck are known a priori, it is logical to assume that the computed vector $\boldsymbol{V}$ = {15.3, 109.0, 0.57, 1.9, 1.1} probably provides a better estimate for the optimum values than the approximate vector shown in Table 5.1 (section 5.1 of this Chapter). The optimum parameter values obtained with the modified version of DSM are almost identical to the values obtained with its regular version. This important result clearly indicates that the performance of the proposed modified version of the algorithm does not degrade on the average, in comparison with the performance of the regular version of DSM.

Sample comparative results for the number of fitness evaluations for the regular and modified versions of DSM are presented in Table 5.4 (the averaged values), and in Figure 5.14. For all test images, the standard values of the DSM coefficients require the most

fitness evaluations L_E, varying from 4655 (for a wing) to 5204 (for a boat), over 100 runs. The use of the response coefficients in the local DSM search results in the significant reduction of the number of evaluations, across all its measures: the mean, the standard deviation, the maximum, the minimum, and the sum, over 100 runs. The range of the reduction rate varies from 27.2% (for a palm) to 44.9% (for a ship), which constitutes significant savings in the computational cost of the local DSM search. Moreover, Figure 5.14 shows that the reduction occurs virtually across all sample runs, and not just on the average.

Image	**Value**	**No. of evaluations**		**Reduction rate, %**	**Min. fitness value**	
		Reg	**Rsps**		**Reg**	**Rsps**
Ship	**Mean**	49	27	44.9	0.041	0.041
	St. dev	8.0	6.0	25.0	0.002	0.002
	Max	75	43	42.7	0.047	0.046
	Min	28	13	53.6	0.038	0.038
	Sum	4911	2704	44.9	--	--
Wing	**Mean**	47	31	34.0	0.002	0.002
	St. dev	6.3	5.0	20.6	0.000	0.000
	Max	65	43	33.8	0.002	0.003
	Min	30	20	33.3	0.002	0.002
	Sum	4655	3101	33.4	--	--

Table 5.4: Number of fitness evaluations and minimum fitness values, for the regular version of DSM (Reg) and its response-enhanced modification (Rsps)

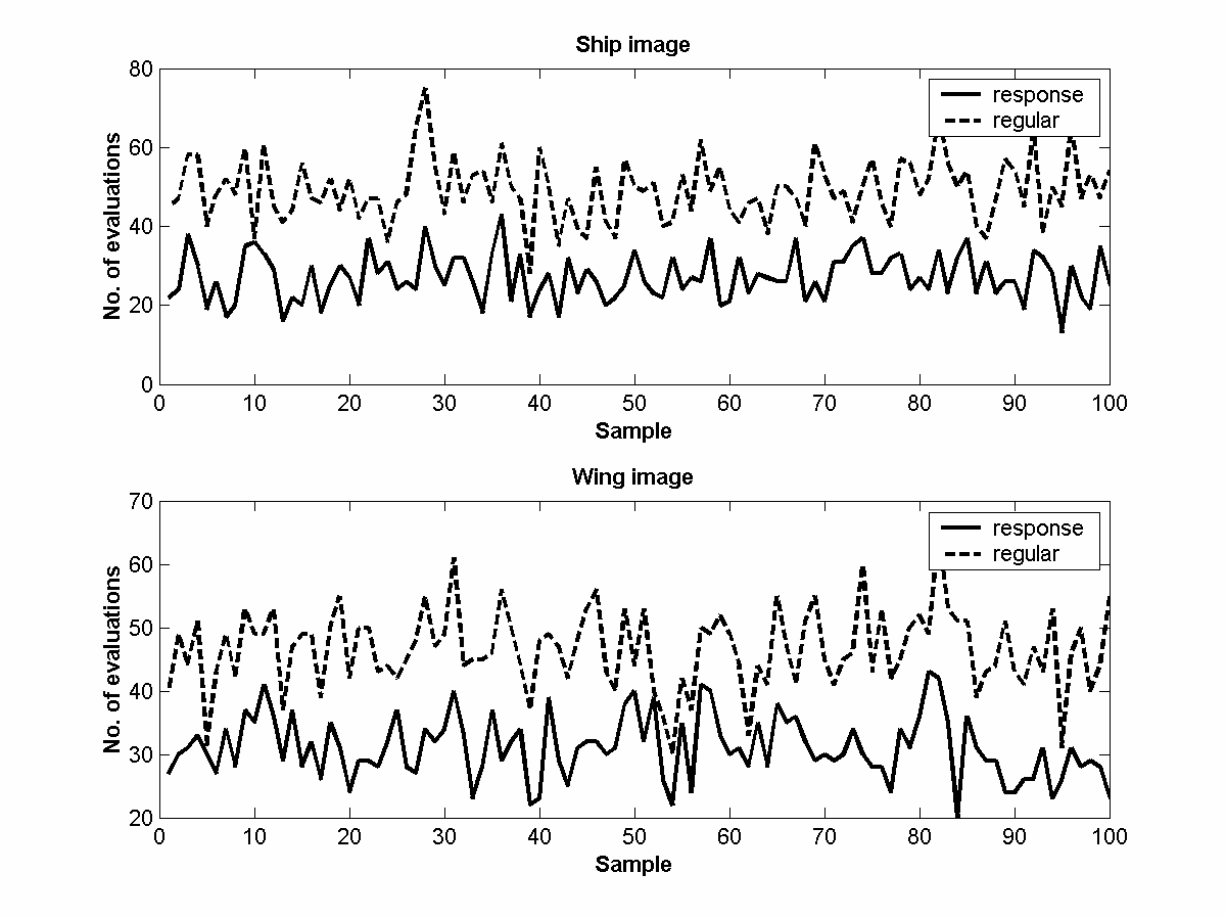

Figure 5.14: Number of fitness evaluations for the regular DSM and its response-enhanced modification

An interesting side effect of the response-enhanced DSM search is the decrease of the variance of the number of fitness evaluations ranging from 13.1% (for a palm) to 34.4% (for a truck). It is reasonable to assume that some smoothing occurs, due to the averaging operation, when local responses are computed.

Computational experiments with a number of images show that the adaptive control mechanism in the local DSM search based on Image local response performs well; it can significantly reduce the number of function evaluations during the DSM search.

However, the reduction comes at the cost of a large number of evaluations of local responses, at different points of the image. The amount of computations can be further reduced if the response surface of the entire image is computed before the evolutionary search starts. A self-organizing network (SON) [174] can be used then, in order to classify different image domains according to their responses, and to build the response map of the image. The algorithm utilizes the basic concepts of SON [174], and works in the pre-processing stage, in the following way.

Step 1: Response matrix M_R of the desired granularity of the image is built for every component of the parameter vector $\boldsymbol{V}$. The corresponding elements of the matrices are averaged over the number of parameters, to obtain one matrix of the average local responses of the image.

Step 2: A self-organizing network with a 2-dimensional lattice (i, j) is built, and initial random values of weights w_{ij} are assigned to its nodes.

Step 3: Every averaged response value R_P is presented to the network, in order to find the weight W closest to the input value, as follows:

$$W = \min\left\| R_P - w_{ij} \right\| \text{ , for all nodes } (i, j) \text{ of the network.} \tag{5.17}$$

Step 4: The weight vector W and its neighbor nodes are iteratively adapted using the following expression:

$$w_{ij}^{new} = w_{ij}^{old} + \varepsilon * (R_P - w_{ij}^{old}) * \exp\left(- \rho * \left\| w_{ij}^{old} - W \right\|^2 \right), \quad (5.18)$$

where i and j range over the neighborhood of W, ε is the learning rate, and ρ is inversely proportional to the size of the neighborhood [174]. The learning rate and the size of the neighborhood are gradually reduced, over the course of the SON iterations.

Step 5: Once the network has been trained, the response map RM_R of the image is built. Different domains of the image are classified by assigning them to the node that has the weight closest to the average response values for the domains.

Step 6: During the evolution and local DSM correction, the response values R_P and R_Q used for computing the coefficients C_{PQ} in Formula (5.16) are retrieved from the corresponding nodes of the network. The network is used here as a compact lookup map of Image local response, thus providing significant reduction in the amount of computations required for evaluating the response coefficients.

The same image sets presented in Figure 5.1 (section 5.1 of this Chapter) are tested, in order to show that the use of the SON response analysis helps reduce the number of function evaluations during the local correction with DSM. Figure 5.15 shows sample 3-D views of the response maps, for the ship and wing images, computed with their respective self-organizing networks.

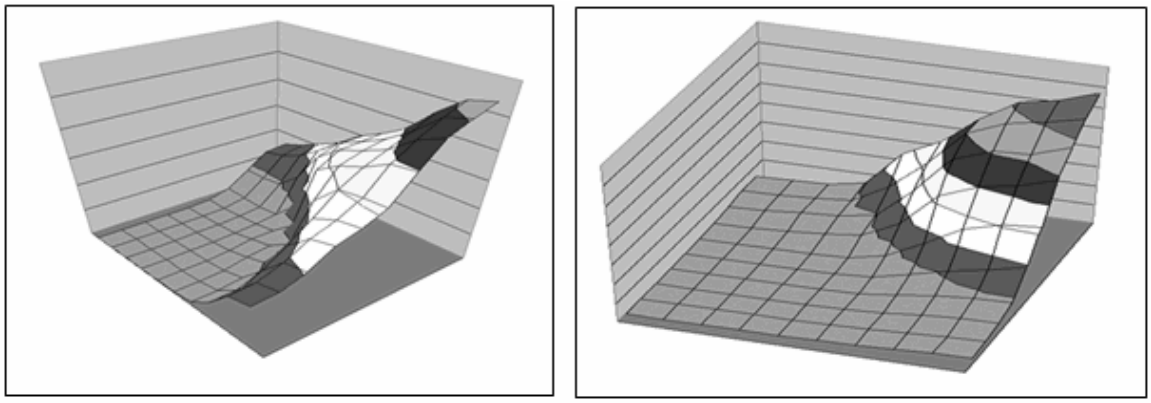

Figure 5.15: Sample 3-D view of the response map: a ship (left) and a wing (right)

Sample results of the test runs are presented in Figure 5.16, where DSM stands for standard algorithm, response – for using local response coefficients, SON – for using SON response map. For all test images, the standard values of the DSM coefficients require the most function evaluations, e.g., for a wing L_E = 48.9 (the mean value). The use of the actual response coefficients C_{Pn} computed for every point participating in the DSM search leads to a significant reduction in the number of evaluations, L_E = 28.3 (the mean value for a wing). This number constitutes the 42% reduction of the computational cost, without the loss of the quality of the final solution. These two cases, the standard and response-modified DSM coefficients, provide the upper and lower bounds for the number L_E. In spite of the replacement of the actual local responses with their approximate values retrieved from the SON-based response map, the SON-based search shows reduction by 41%, with L_E = 29.0 (the mean value for a wing), which lies between the two bounds.

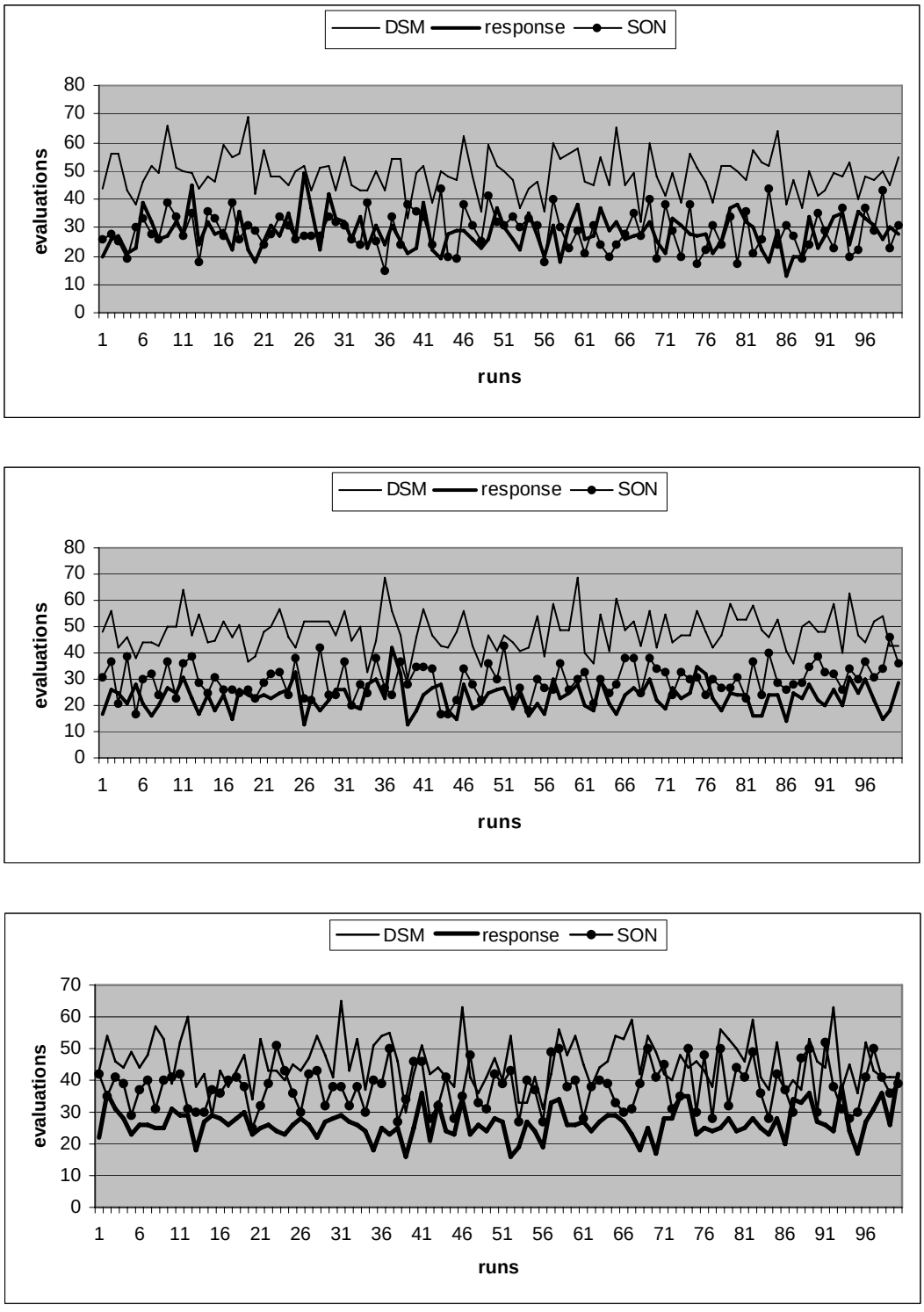

Figure 5.16: Number of function evaluations over 100 runs: a wing (top), a ship (center), and a palm tree (bottom)

Image	Value	No. of fitness evaluations			No. of response evaluations		
		DSM	Response	SON	DSM	Response	SON
Wing	**Mean**	48.9	28.3	29.0	-	-	-
	St. dev.	7.14	6.25	6.72	-	-	-
	Sum	-	-	-	0	28290	16384
Ship	**Mean**	48.2	23.3	29.8	-	-	-
	St. dev.	7.26	5.15	6.23	-	-	-
	Sum	-	-	-	0	23330	16384
Palm	**Mean**	45.5	26.5	37.8	-	-	-
	St. dev.	7.54	4.88	6.92	-	-	-
	Sum	-	-	-	0	26470	16384

Table 5.5: Comparative results for the number of fitness evaluations and the number of response evaluations

Table 5.5 summarizes the results of the experiments, where DSM stands for standard algorithm, Response – for response-enhanced algorithm, and SON – for SON and response-enhanced algorithm. Using the actual local responses gives maximum savings in computations ranging between 42% and 52%, for the tested images. The downside of this technique is that it requires a large number of response evaluations L_E growing proportionally to the number of iterations and the number of chromosomes participating in the DSM search. The use of the SON-based response map requires only one-time evaluation of the response matrix for the entire image, in order to build its response map. The number of response evaluations is constant; it does not depend on the number of

iterations, nor does it depend on the size of the best chromosome pool. The use of SON results in the reduction of the number of response evaluations from 28290 to only 16384 (for a wing), which constitutes a 42% reduction. For all tested images, the number of response evaluations ranges between 58% and 70% of the number of evaluations for the actual responses. However, the replacement of the actual response values with their approximations from the SON response map might result in the greater number of fitness evaluations required during the evolutionary search. Overall, the maximum total savings in computations range between 17% and 41%, for the tested images.

In summary, Image local response can be used to adaptively control the DSM simplex by applying the contraction transformation to the vector of the standard DSM coefficients. This technique makes the walk of the DSM simplex in the search space correlated with the local properties of the gray values surface at different points of the image. The computational experiments with the grayscale images provide the experimental support and justification of the analytical model of Image local response and its utilization for reducing the computational cost of the local DSM search. The results of the test runs for the regular and response-enhanced DSM search, with the (±10%) parameter range around the optimum show that for all tested images, the use of the response coefficients leads to a significant reduction in the number of fitness evaluations. The reduction rate varies between 27.2% and 44.9%, for different test images. Moreover, the quality of the solution does not degrade, in comparison with the regular version of DSM. The results clearly indicate that the proposed response-enhanced algorithm of local DSM search provides

significant reduction of computational cost without the loss of the quality of the final solution.

The reduction in the number of function evaluations comes at the price of computing the local response coefficients at multiple points of the image during the runtime. To reduce the amount of computations, the response matrix for the entire image can be computed in the pre-processing stage, before the evolutionary search begins. A self-organizing network can be utilized then, to classify different points according to their response values, and to construct an adaptive and compact response map of the image. During the main run of the evolutionary procedure, the map is used as a compact lookup table allowing algorithm to retrieve the approximate values of the response coefficients. The use of the SON response map helps offset the large number of response evaluations associated with the local response model and adaptive control mechanism of the local search, and reduces the number of function evaluations up to 41%, for the tested images.

5.6. Reduction of Search Space during Selection and Crossover

The operator of selection in Evolutionary algorithms is responsible for selecting parental chromosomes that will participate in producing the offspring for the next generation. Selection mechanism plays a major role in the evolutionary search guiding it in the direction toward the fitter fraction of the population. All implementations of selection mechanism are aimed at creating more copies of higher-fit individuals, i.e., solutions with lower values of fitness function, in the case of minimization problem.

Selection, while favoring the fitter solutions, does not create new chromosomes, though. The creation of new solutions is the responsibility of the operators of crossover and mutation. Crossover does not produce new values of the components of the parameter vector, i.e., the values of *DX*, *DY*, *θ*, *SX*, and *SY* (in the case of the 5-dimensional affine image mapping); it only creates new combinations of the existing values taken from the parental chromosomes. On the contrary, mutation actually produces new values of the components, thus exploring new sub-areas of the search space. Acting together, crossover and mutation supply the population pool with new candidate solutions.

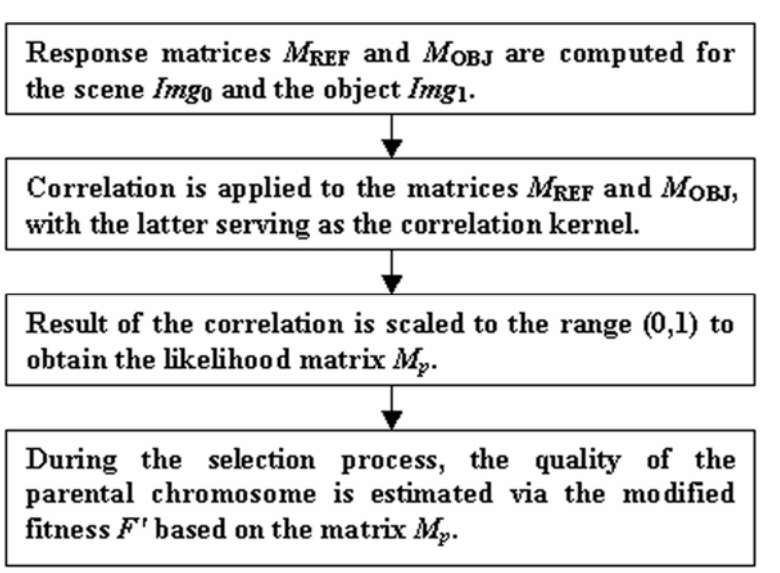

Figure 5.17: Algorithm of using Image local response in the operations of selection and crossover

Image local response can be used to limit the creation of new candidate solutions to the sub-areas of the search space that most likely might contain the optimum solution. The

modified procedure of selection and crossover shown in Figure 5.17, utilizes the likelihood matrix M_p based on Image local response, in the following way.

Step 1: Images Img_0 and Img_1 are covered with the regular grid of the specified granularity, and the response values R_P are computed for every point $P(i,j)$ of the grid, where $i = 1,\ldots,I$, $j = 1,\ldots,J$; and I and J are the respective numbers of rows and columns in the grid.

Step 2: The computed response values form response matrices M_{REF} and M_{OBJ} of the scene Img_0 and the template Img_1, respectively. The size of each response matrix corresponds to the size of the corresponding image, and the element $m(i,j)$ of the response matrix equals the response R_P at the point $P(i,j)$ of the image.

Step 3: The operation of cross correlation is applied to the matrices M_{REF} and (possibly, scaled down) M_{OBJ}, such that the latter serves as the correlation kernel. In order to increase the signal level, the operation of cross correlation can be repeated, with M_{OBJ} rotated e.g., by 90°.

Step 4: A threshold can be applied to the gray values of the correlated image obtained in step 3, in order to eliminate the background areas, and to emphasize the areas occupied by the objects presented in the scene.

Step 5: The result of the correlation is scaled to the range (0,1). After scaling, the resulting matrix M_p, of the size of the image Img_0, serves as the likelihood matrix applied during the process of selection and crossover, as discussed below.

Since M_p is the result of cross correlation between the response matrices of the images, it accentuates the sub-regions in the scene, where the template most likely can be found. The value of the element $p(i,j)$ of the matrix M_p corresponds to the likelihood of locating the optimum solution $\boldsymbol{V}^*$ at the point $P(i,j)$ of the scene. The sub-areas with the higher probability of the optimum solution will have higher values of $p(i,j)$. For example, if the image Img_0 is partitioned into sub-regions corresponding to the objects in the scene and surrounding background, then the sub-regions corresponding to the background will have nearly zero likelihood of the optimum solution, i.e., the values $p(i,j)$ in these sub-regions will be close to zero. On the contrary, sub-regions corresponding to the objects will have fairly high probability values $p(i,j)$ in M_p. Matrix M_p can be thought of as a mask applied to the image Img_0, so that the background areas will be suppressed, while the sub-regions corresponding to the objects will have a high likelihood of locating the optimum solution.

During the evolutionary search, the parental chromosomes are selected in accordance with the modified quality F' estimated with the following expression:

$$F' = F \cdot \exp[(1-p)^2], \tag{5.19}$$

where F is the actual fitness value, and p is the probability corresponding to the chromosome's entry $p(i,j)$ in the likelihood matrix M_p. The exponential term in (5.19) plays a role of a penalty for the chromosome's being in the sub-region of the low

likelihood of the optimal solution: the penalty, and the corresponding modified fitness value F' exponentially grow, as the probability p decreases.

Sample sets of 2-D grayscale images are tested, in order to evaluate the proposed algorithm. The first test set is shown in Figure 5.18. The set consists of the 256×256-pixel reference image (Figure 5.18, left) [71], with the indicated location of the object, and the 130×90-pixel, transformed and distorted (SX ≠ SY) image of the object (Figure 5.18, right). The optimum values of the components of the 5-dimensional parameter vector $\boldsymbol{V}^*$ = {*DX, DY, θ, SX, SY*} are $\boldsymbol{V}^*$ = {31, 141, 1.57, 4.0, 2.0}.

Figure 5.18: Reference image (left) with the indicated location of the object, and the transformed image of the object (right)

The optimization problem is solved with the modified HEA model discussed in Chapter 4. The population size N_P = 133 is constant throughout the entire algorithm run. The

initial population is generated at random from the entire search space, with the step size 2 pixels along the *x* and *y* axes, 5.7° along the rotation angle, and 0.1 along the scaling factors. The best 10 chromosomes of the current population undergo local improvement with the two-phase cyclic local search. The best offspring produced after crossover, and the best mutants created using mutation with memory, replace the worst fraction of the current population, to form the new generation.

The recognition of the image of a boat in Figure 5.18 (left) is a complex computational problem. The small boat object is located in the upper half of the scene, which is significantly cluttered with bushes and trees. The object does not appear to have noticeable features that would single it out against the background. What makes the recognition task even more complex for the regular HEA is the presence of the vast water area with the high-intensity reflections, in the lower half of the image. Simple analysis shows that, on the average, the difference between the pixel values of the object and the pixel values in the water area are smaller than the difference between the pixel values of the object and the pixel values in the upper part of the scene image. It means that the chromosomes placed in the water area will have lower fitness values *F* than the chromosomes placed in the upper part of the scene image (on the average).

Following the evolutionary strategy that favors chromosomes having lower fitness values, the algorithm will focus the search on the water area of the image, and will overlook the optimum solution located in the upper part of the image, in the bush area. In the particular case of the boat image, the regular algorithm is terminated after 90

generations, with the search being trapped at one of the local minimum positioned in the lower half of the image, in the water area. This type of problem is known as a deceptive function in the EAs theory. For the deceptive function, the intermediate partial solutions have low fitness values misleading the algorithm, and directing the search toward the areas that are away from the optimum solution.

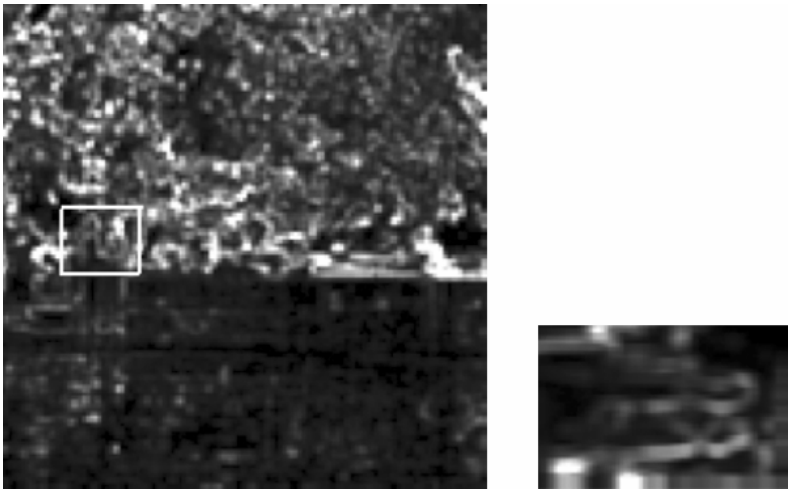

Figure 5.19: Average responses of the reference (left) and object (right) images

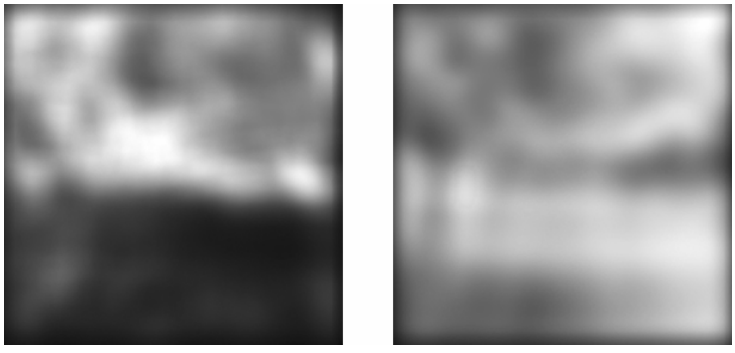

Figure 5.20: Cross correlation of response matrices in the response (left) and image (right) spaces

Figure 5.19 shows the averaged local responses of the reference image (Figure 5.19, left), and the object image (Figure 5.19, right). The result of the cross correlation of the response matrices M_{REF} and M_{OBJ} is shown in Figure 5.20 (left). The operation of cross correlation is applied, with the correlation kernel M_{OBJ} scaled down by the factor 4.5 and rotated by 0° and 90°. For comparison, a similar correlation procedure is applied to the original images shown in Figure 5.18. The result of their correlation in the actual image space (i.e., using the actual image gray values) is shown in Figure 5.20 (right). Both images in Figure 5.20 can be viewed as the probability matrices, where the sub-regions having higher intensity values correspond to the image regions with a higher likelihood of the optimum solution. One can immediately see that the correlation of the response matrices effectively reduces the probability of search for the global solution in the water area of the reference image. On the contrary, the correlation of the original images displays high probabilities of finding the global optimum solution across the entire area of the reference image.

Figure 5.21: Mapping results for the response-based algorithm after 24 generations (left), and for the regular algorithm after 90 generations (right)

As can be seen in Figure 5.20 (left), the correlation of the response matrices effectively reduces the probability *p* of selecting the parents in the water area, where the value of *p* is close to zero. Consequently, the penalty and the modified fitness value *F′* computed according to Formula (5.19) for the chromosomes located in this area exponentially grow, which effectively reduces their chance to be selected as parents. The scope of selection and recombination is reduced, therefore, to the upper half of the scene, where the boat object is actually located. The algorithm is able to find the object after 24 generations, with the optimum parameter vector $\boldsymbol{V}$ = {33, 144, 1.61, 3.71, 2.1} - see Figure 5.21 (left). The vector of the absolute differences between the calculated and exact parameter values (on a scale of 100 percent) is small, and equals $\boldsymbol{E}$ = {6.5%, 2.1%, 2.5%, 7.3%, 5.0%}.

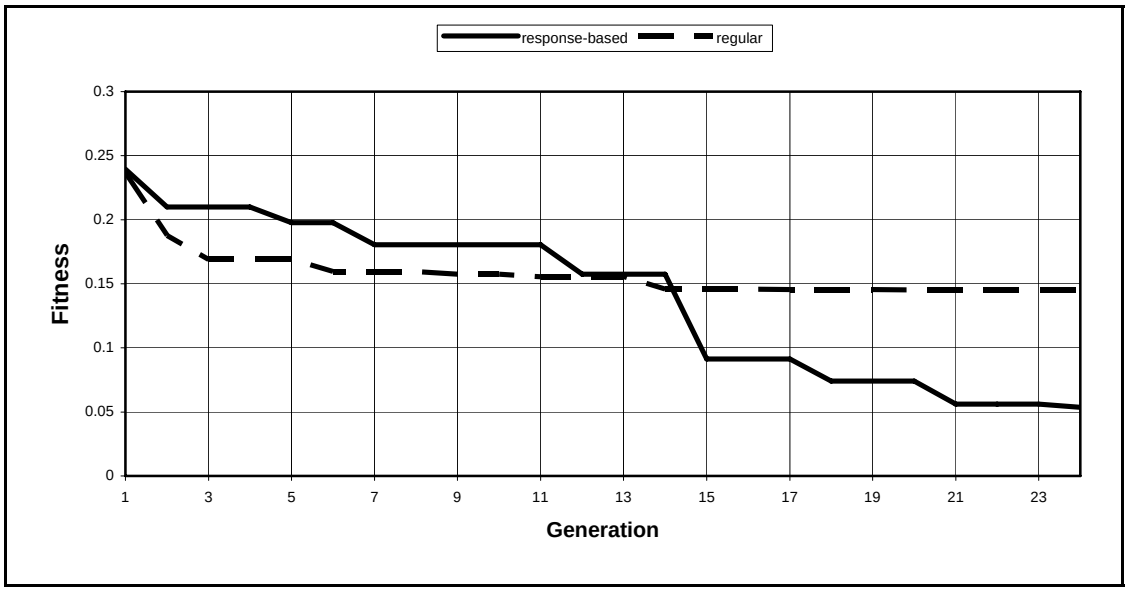

Figure 5.22: Comparative performance of the response-based and regular algorithms

The comparative performance of the regular algorithm and its response-based modification for the boat image is shown in Figure 5.22. As one can see, the regular

version leads at the beginning of the search while exploring the low-fitness water area, but stalls in one of the many minima located in this area. The modified version utilizes the probability matrix shown in Figure 5.20 (left), to avoid the deceptive area, and successfully finishes the search with the optimum configuration of the parameter vector.

The second image set shown in Figure 5.23 includes the 256×256-pixel scene containing several objects (Figure 5.23, left) [70], and the 170×128-pixel image of the sought object (Figure 5.23, right). The optimum value of the parameter vector is $\boldsymbol{V}^* = \{150, 212, 1.57, 4.14, 2.07\}$. The image of a wing is another interesting and computationally hard recognition problem. Six projections of the same object are treated here as six different objects in the 2-D scene. The objects have similar shapes, but only one of them is the actual (optimum) solution. The purpose of the experiment is to test the ability of the algorithm to distinguish between the similar objects. The algorithm has to identify the correct object, via investigating the possible combinations of the global and local transformations for all objects in the scene, and via detecting the transformation and the object that yield the minimum fitness value.

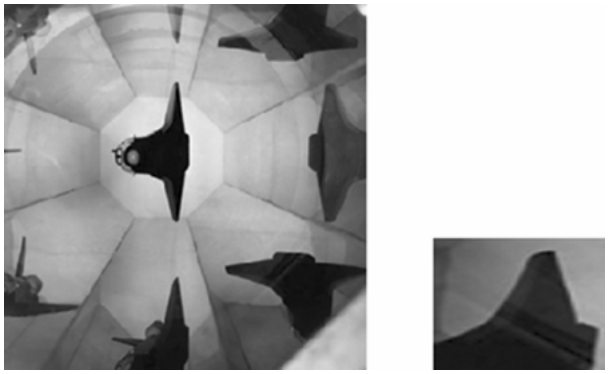

Figure 5.23: Original scene (left) and the image of the sought object (right)

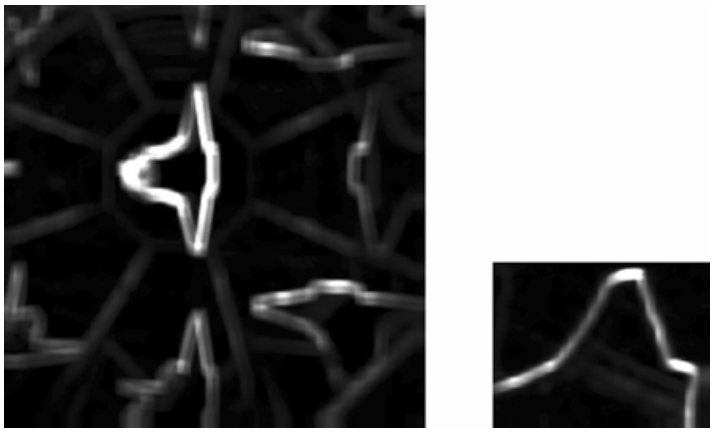

Figure 5.24: Average responses of the original scene (left) and the sought object (right)

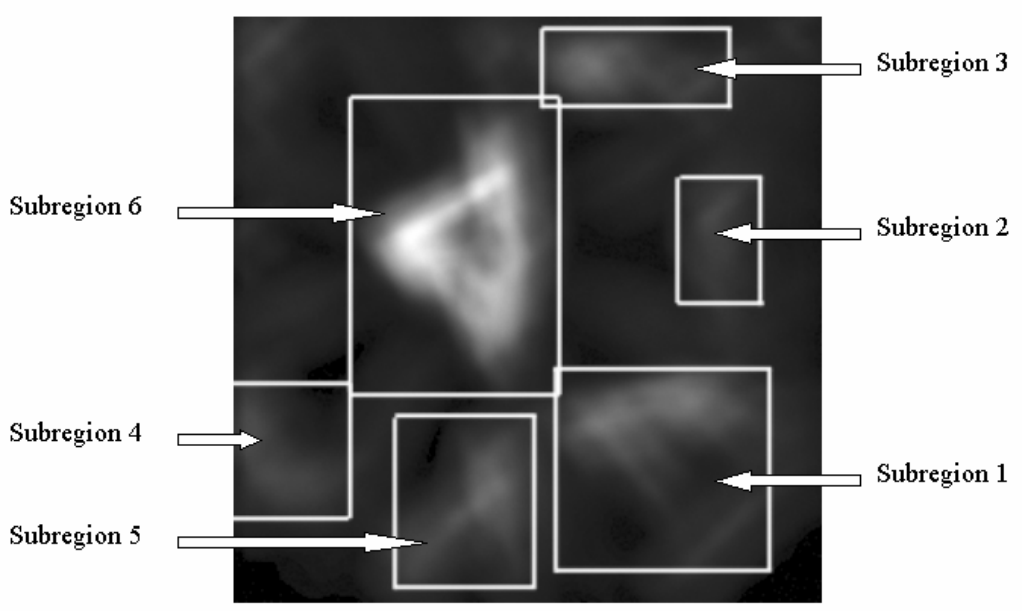

Figure 5.25: Cross correlation in the response space, with the indicated sub-regions of the potential object match

Figure 5.24 shows the average local responses of the scene (Figure 5.24, left) and the sought object (Figure 5.24, right). The result of the cross correlation of the response matrices M_{REF} and M_{OBJ} is shown in Figure 5.25. The correlation is applied twice, with

the correlation kernel M_{OBJ} scaled down by the factor 6.5 and rotated by 0° and 90°. Six indicated sub-regions, labeled 1 through 6, correspond to the objects in the original scene; they are obtained by applying the 40-level threshold to the gray values (ranging from 0 to 255). The image in Figure 5.25 can be viewed as the probability matrix M_p defining the likelihood of finding the solution in different sub-regions of the original scene. The correlation of the response matrices effectively divides the search space into six subpopulations corresponding to the six sub-regions.

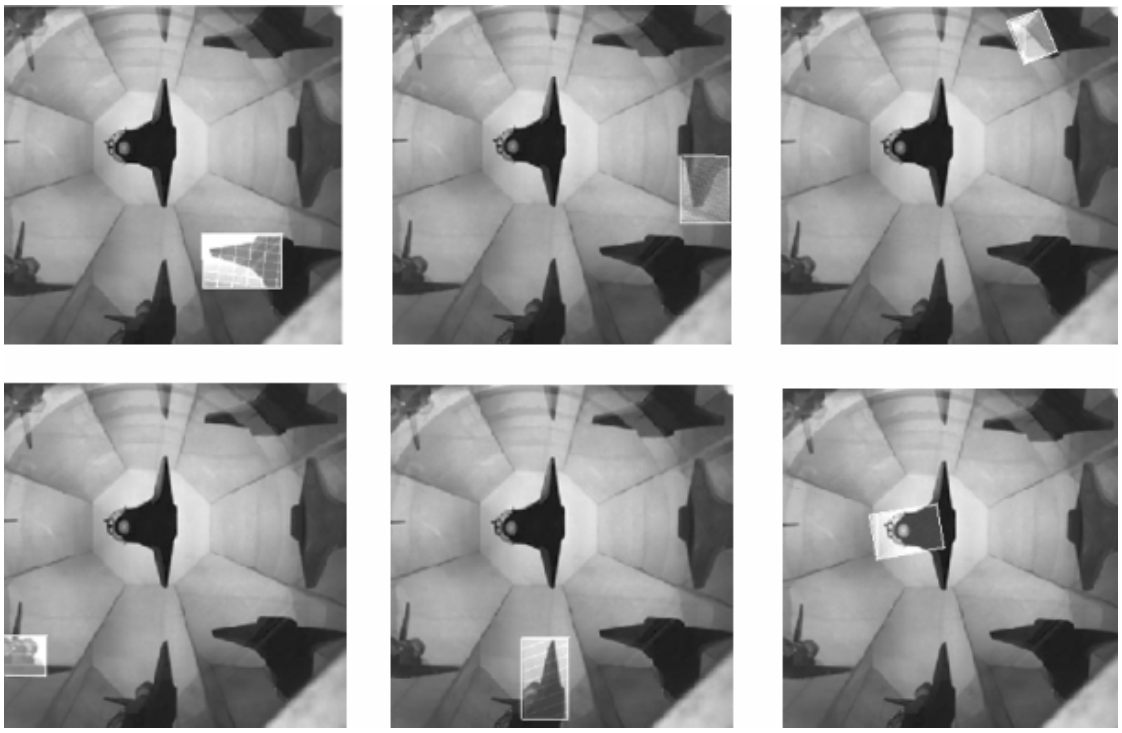

Figure 5.26: Results of image mapping for the sub-regions 1 through 3 (upper row, left to right), and 4 through 6 (lower row, left to right)

Figure 5.26 presents the results of the image mapping, when each of the six subpopulations is processed independently. The search is terminated after 15 generations,

for each subpopulation. The lowest value of the fitness function $F_1 = 0.00406$ is obtained after 7 generations, for the sub-region 1 corresponding to the correct object, with the respective parameter vector $\boldsymbol{V_1} = \{149, 212, 1.567, 4.15, 2.08\}$, and the error vector (on a scale of 100 percent) $\boldsymbol{E} = \{0.67\%, 0.0\%, 0.0\%, 0.22\%, 0.48\%\}$. Figure 5.27 shows the comparative mapping results for the sub-regions 1 through 6. The closest to the winner is the sub-region 2, with the minimum value of the fitness function $F_2 = 0.05626$, which is by the order of magnitude larger than F_1. The significant difference between the two leading solutions clearly marks the object in the sub-region 1 as the correct object. The algorithm is able to discriminate between the objects having a high degree of similarity, and to successfully identify the correct object.

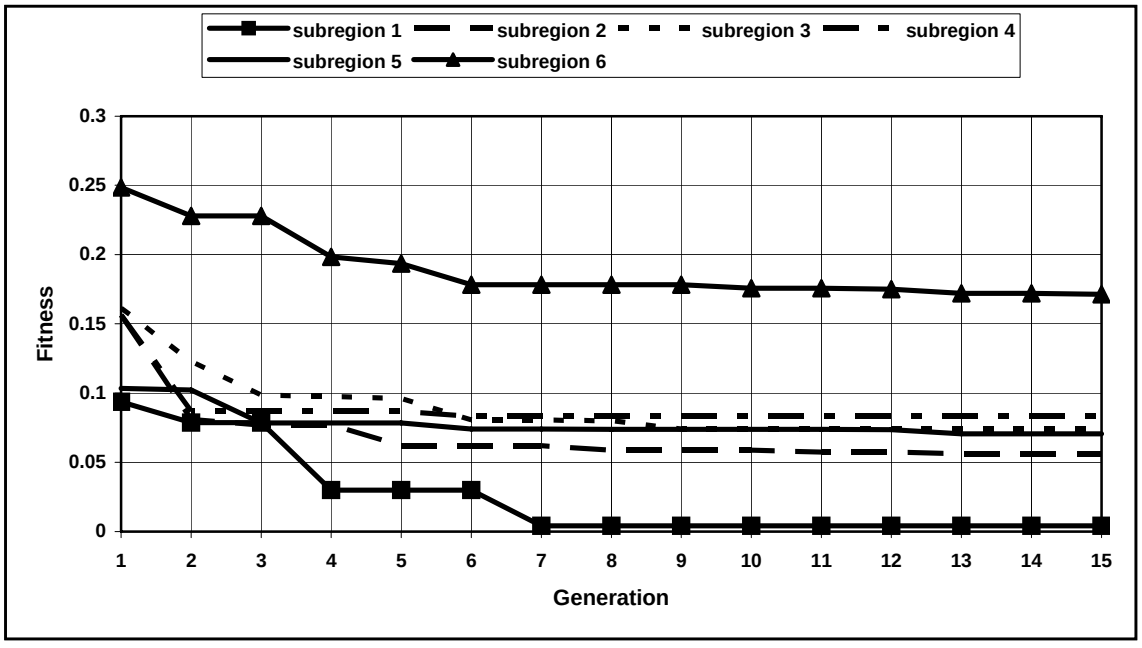

Figure 5.27: Comparative performance of the subregion-based mapping

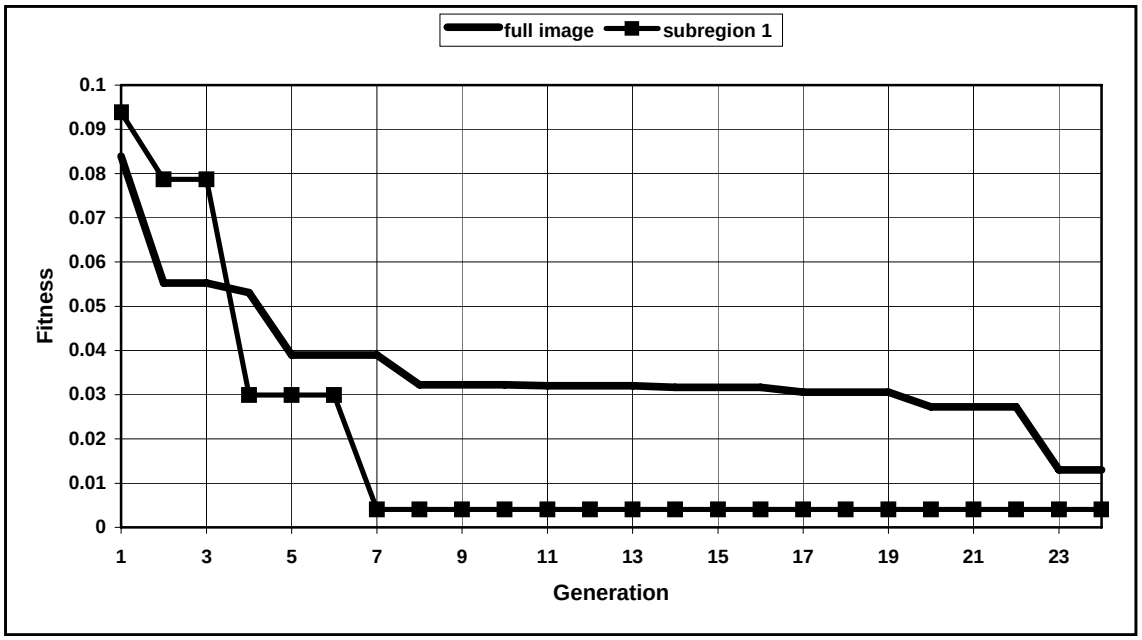

Figure 5.28: Comparative performance of the subregion 1 and full image mappings

In the absence of the response analysis, the regular algorithm searches the entire image using a single population. It is able to find the correct object after 23 generations, with the minimum fitness value $F = 0.01303$, the optimal parameter vector $\boldsymbol{V} = \{154, 212, 1.57, 3.96, 2.29\}$, and the error vector $\boldsymbol{E} = \{2.7\%, 0.0\%, 0.0\%, 4.3\%, 10.6\%\}$. The comparative performance of the response-based and regular models of the algorithm is presented in Figure 5.28. Although the regular algorithm leads at the beginning of the search, the proposed response-based modification converges faster to the optimum mapping. As one can see, a significant speedup (7 vs. 23 generations) is achieved when the response analysis is used at the pre-processing stage, in order to identify the prospective sub-regions of the search, and to process them concurrently as independent subpopulations.

5.7. Multi-Resolution Correlation of Image Local Responses in Object Recognition

Computing cross correlation between images either in spatial, or in frequency domain remains one of the commonly used approaches in object and target recognition. The maximum value of the correlation function corresponds to the properly aligned images, i.e., the image Img_T of the target is correctly recognized in the image Img_S of the scene. The use of the correlation as recognition technique is limited to the cases where:

- images are subject to similarity transformation including translation, rotation, and isotropic scaling, which excludes any noticeable distortion of the images;
- the space of all feasible geometric transformations is relatively small, which allows for conducting an exhaustive search for the proper values of the transformation parameters, with the acceptable level of accuracy.

When, in addition to translation, images are subject to rotation and scaling, usually an exhaustive search is performed in the space of the feasible parameter values, in order to find the maximum value of the correlation function corresponding to the correct match of the images. If images are somehow distorted, correlation usually does not produce reasonably useful information, in the form of a peak corresponding to the correct match, as in the case of a simple translation. It rather produces a relatively homogeneous stochastic distribution of correlation levels across the entire image, e.g., as in Figure 5.20 (right), in section 5.6 of this Chapter.

Image local response significantly reduces the amount of information presented in the image eliminating unimportant pixel areas, and retaining only the areas that are essential

for the sufficient image representation. One can expect that the operation of cross correlation performed on image response matrices might result in the distribution of correlation levels that can potentially identify the most likely areas of interest in the entire parameter space during the image mapping. The techniques that utilize correlation of image responses for such an identification and experimental results in their support are presented in this section.

5.7.1. Image Response and Multi-Resolution Correlation in Object Recognition

Image local response can be utilized for multi-resolution automatic object recognition. The recognition task is re-formulated as a nonlinear global optimization problem, i.e., the search for a proper transformation $A(\boldsymbol{V})$ that provides the best match between the images of the object and the scene. Given the images of the scene and the targeted object located in the scene, one can repeatedly apply response analysis on different resolution levels (zooming in on the scene), in order to find the region of interest (ROI) containing the object in the large-scale image of the scene.

On each resolution level, the response matrices are computed for the ROI and object images. Cross correlation of the response matrices built for ROI and object outlines potential locations of the latter. Once the locations are identified, the algorithm zooms in on the found locations. The response-based algorithm for multi-resolution object recognition can be formulated now in the following way.

Step 1: Matrices M_T and M_S of Image local response are computed for the object and scene images, on the current resolution level.

Step 2: Cross correlation is applied to the response matrices M_T and M_S, in order to identify the regions of interest that might potentially contain the object. During the correlation, the response matrix of the object serves as the correlation kernel. The operation of cross correlation can be repeated, with the rotated (e.g., by 90°) kernel, in order to amplify the signal.

Step 3: Once the regions of interest have been identified, the algorithm zooms in on the found locations, and the hybrid EA procedure is applied to the response matrices M_T and M_S; it attempts to minimize the least squared difference of the values of the response matrices corresponding to the images, thus searching for the correct parameter vector $\boldsymbol{V}$ for the targeted object, with the reference to ROI.

Step 4: The procedure that includes the computation of the response matrices, their cross correlation, and object recognition with the hybrid EA procedure, is repeated until the resolution level is reached that makes possible the correct recognition of the targeted object.

In order to test the approach for object recognition based on the multi-resolution use of Image local response, computational experiments are conducted on a set of 2-D grayscale images. Figure 5.29 shows two images of the same scene at the different resolution

levels; the targeted objects are indicated with the arrows in the left image; the right image corresponds to the zoomed-in white-colored frame in the left image. The left image in Figure 5.29 has the dimension 512×512-pixel [75]; two identical objects are indicated with the white arrows. The right image in Figure 5.29 is the rotated and zoomed-in region of interest (ROI) corresponding to the white-colored frame on the left image [74].

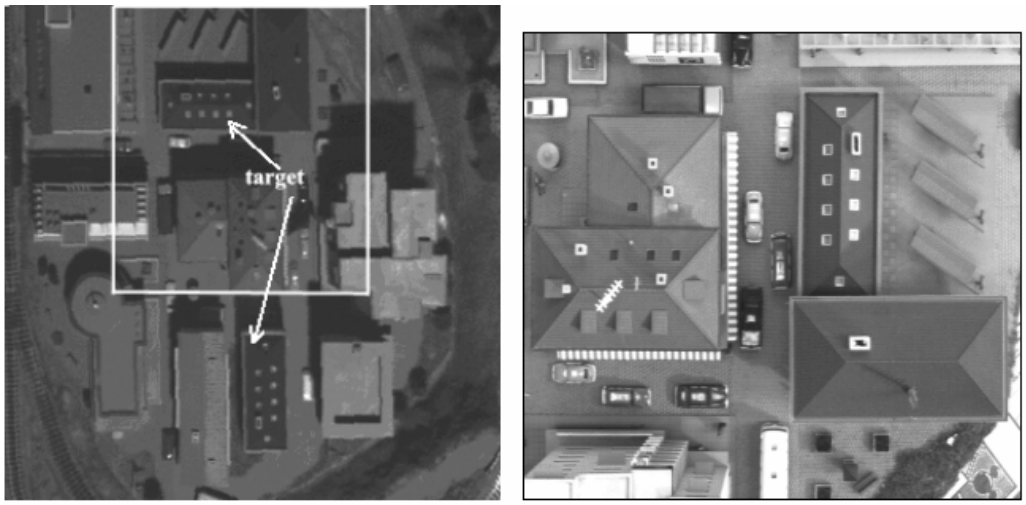

Figure 5.29: Two images of the same scene at the different resolution levels

The response matrices of the scene M_{S1}, and the object M_{T1} are computed; they are shown in Figure 5.30. The operation of cross correlation is applied to the response matrices of the both images, where the object M_{T1} serves as the correlation kernel. Images are correlated twice, with the correlation kernel M_{T1} rotated by 0°, and 90°. Figure 5.31 (left) shows the result of the correlation, with the indicated sub-regions 1 through 4 obtained by applying the 50% threshold to the pixel values. The sub-regions 1 through 4 effectively identify the areas that potentially might contain the targeted object, and have to be zoomed-in for further recognition. Figure 5.31 (right) shows the corresponding sub-

regions in the scene; the highlighted sub-regions 1 and 4 actually contain the objects. As one could expect, the correlation of the actual images (i.e., the correlation of the actual image pixel values) of the scene and the object shown in Figure 5.32 cannot clearly identify the areas of the potential object locations.

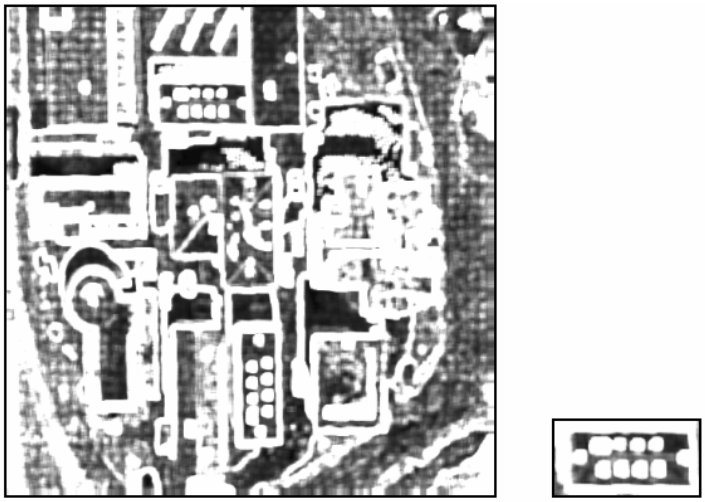

Figure 5.30: Response matrices of the scene (left) and the targeted object (right)

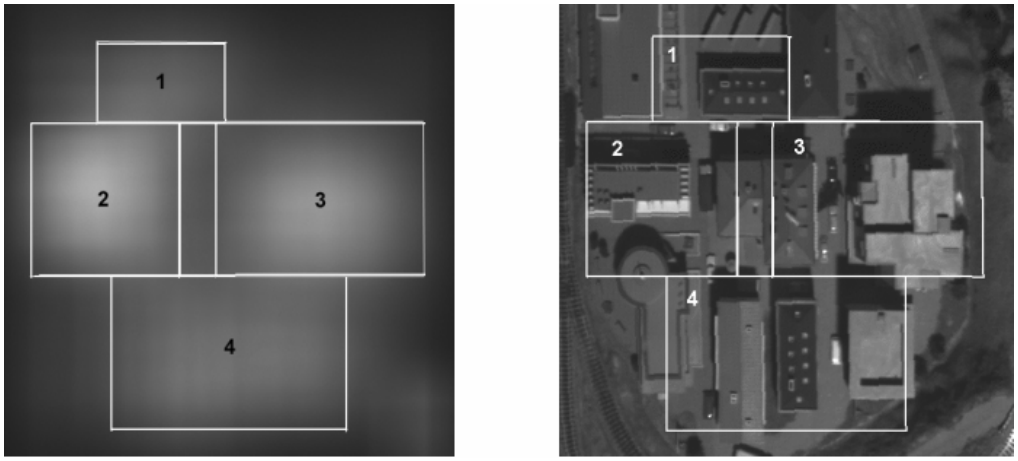

Figure 5.31: Correlation in the response space, with the indicated sub-regions of the potential location of the targeted object (left); the corresponding ROI in the scene (right)

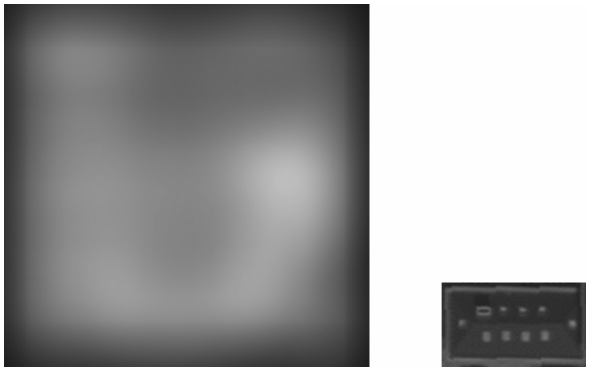

Figure 5.32: Correlation in the actual image space (left); the actual image of the targeted object used as the correlation kernel (right)

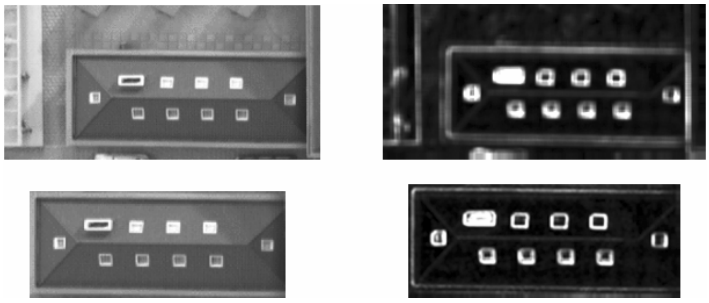

Figure 5.33: Zoomed-in region of interest and its response matrix (top row, left to right); zoomed-in targeted object and its response matrix (bottom row, left to right)

The sub-region 1 is chosen for the object recognition, and its zoomed-in version is obtained from the ROI image (see Figure 5.30, right). Figure 5.33 shows the zoomed-in sub-region 1 (Figure 5.33, top row, left) and its response matrix M_{S2} (Figure 5.33, top row, right), as well as the zoomed-in targeted object (Figure 5.33, bottom row, left), with the corresponding response matrix M_{T2} (Figure 5.33, bottom row, right). Object recognition in the sub-region 1 is formulated as the problem of finding the optimum vector $\boldsymbol{V}^*$ minimizing the difference F between the images. The optimization problem is investigated with the hybrid EA procedure, and the fitness values corresponding to

different parameter vectors are obtained. The optimum mapping in the response space is shown in Figure 5.34.

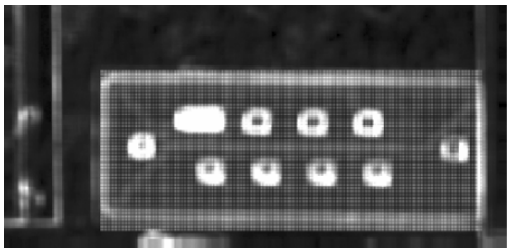

Figure 5.34: Location of the targeted object corresponding to the optimum parameter vector

10% shift	**Parameter values**					**Fitness value**
	DX	***DY***	***θ***	***SX***	***SY***	
None	58	38	0.0	1.5	1.5	0.0283
DX	30	40	0.0	1.5	1.5	0.1050
DY	60	26	0.0	1.5	1.5	0.1642
SX	60	40	0.0	1.65	1.5	0.0771
SY	60	40	0.0	1.5	1.65	0.0862
SX, SY	60	40	0.0	1.65	1.65	0.1000

Table 5.6: Parameters for the optimum solution and the neighboring points shifted by 10% from the optimum point

Table 5.6 gives the comparative results for the parameters and fitness values for the optimum solution and for the neighboring points in the parameter space. Each neighbor has one of the components {*DX*, *DY*, *θ*, *SX*, or *SY*} shifted by 10% from its optimum

value. As one can see, the fitness values computed for the non-optimum parameters are significantly higher than the optimum fitness value $F = 0.0283$. In addition, the fitness values computed for a number of parameter vectors outside the 10% parameter range are also significantly higher than the optimum fitness value; they exceed the value $F = 0.06$. The large difference in the fitness F values, where the minimum value corresponds to the optimum solution, provides the experimental support for the multi-resolution use of Image local response in solving object recognition problem.

5.7.2. Estimating the Scaling Factor with Correlated Image Response

Imaging optimization with Evolutionary algorithms requires estimating the range within which the value of each parameter can vary: the smaller the range, the more efficient the search can be. In a fully automatic recognition, the task of defining parameter ranges has to be handed over to the algorithm. The algorithm proposed in this section repeatedly applies cross-correlation between response matrices on different resolution (i.e., zoom) levels, in order to find the lower and upper bounds of the scaling factor in the parameter vector $\boldsymbol{V}$. Once the range of the scaling factor is identified, a hybrid EA procedure can be applied to the response matrices searching for the correct parameter vector $\boldsymbol{V}$ for the targeted object. Image local response can be utilized to identify the range of the scaling factors SX and SY in the following manner.

The matrices M_{T} and M_{S} of image local response are computed for the respective images Img_{T} and Img_{S} of the targeted object and the scene. The operation of cross-correlation is performed between the matrices for a range of resolutions, to obtain the rough estimates

of the lower and upper bounds of the scaling factors *SX* and *SY*. The normalized cross-correlation function $c(m,n)$ takes its regular form:

$$c(m,n) = \frac{\sum_{\Omega}[r_T(x,y) \times r_S(x+n, y+m)]}{\left(\left(\sum_{\Omega} r_T^2(x,y)\right) \times \left(\sum_{\Omega} r_S^2(x+n, y+m)\right)\right)^{0.5}}, \qquad \{5.20\}$$

where $r_T(x,y)$ and $r_S(x,y)$ are the response values of the images Img_T and Img_S, respectively, and Ω is the area of their overlap [134].

If images Img_T and Img_S have the same values of the scaling factors $SX = SY = 1$, and the other components of the parameter vector $\boldsymbol{V}$ equal to zero, then the maximum value of the correlation function $c(m,n)$ corresponds to the position $\boldsymbol{X} = \{m,n\}$, at which the image Img_T is properly aligned, i.e., the object is correctly recognized in the scene image Img_S. Since this is rarely the case, cross-correlation can rarely be used as a recognition technique in real-world applications. However, it can still be used to obtain the crude estimates of the lower and upper bounds of the scaling factors. As experiments show, the cross-correlation of the response matrices produces a 2-dimensional function $c(m,n)$ whose histogram is dependent on the resolution and is significantly invariant to the rotation of the target. Computing the cross-correlation of the response matrices for a set of successive resolution levels allows the algorithm to identify the range within which the scaling factors can vary. Thus identified range can be used in reducing the size of the search space in object recognition with Evolutionary algorithms.

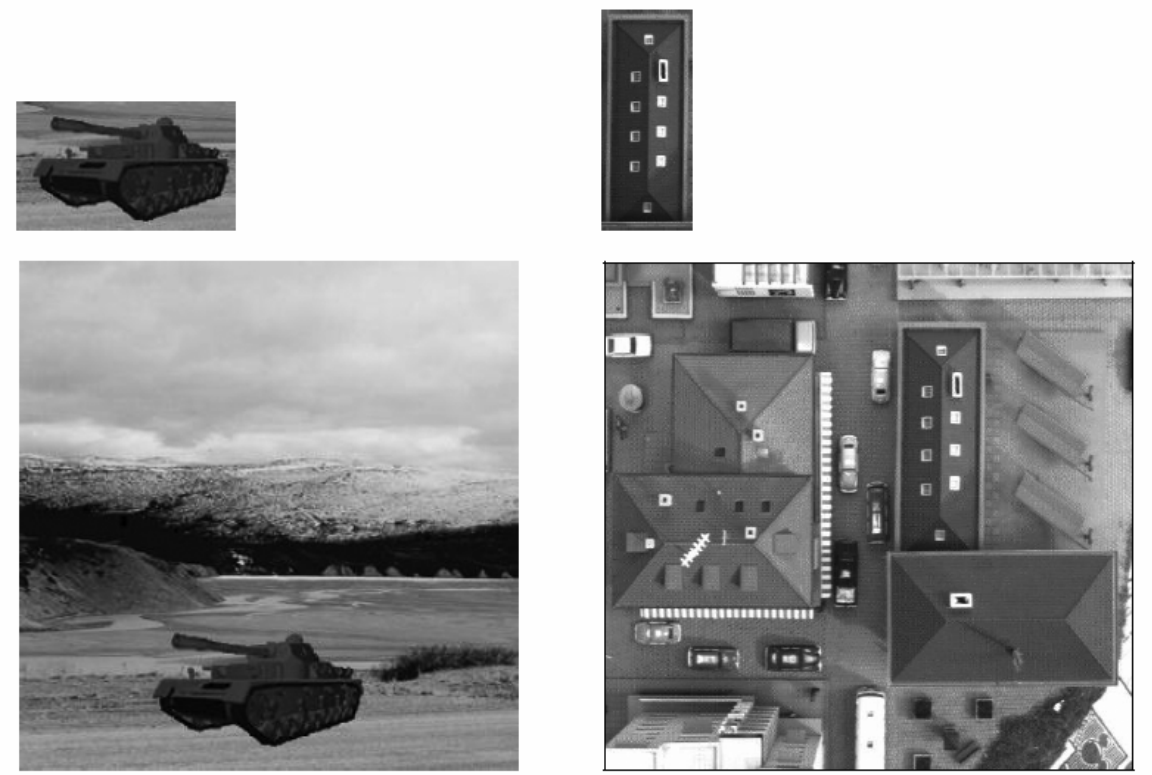

Figure 5.35: Test sets of an object (top row) and a scene (bottom row): a vehicle (left) and a building (right)

In order to illustrate the proposed technique, computational experiments are conducted on two sets of 2-D images shown in Figure 5.35: a vehicle and a building [74]. Each set includes an image Img_T of a targeted object and an image Img_S of a scene containing the object. Figure 5.36 shows the response matrices M_S of the scenes. The cross-correlation of the matrices M_T and M_S is performed for different values of rotation angle of the object θ = (0º, 90º, 180º, 270º), for a wide range of resolution levels $SX = SY$ = (0.1, 0.15, 0.2, 0.25, 0.33, 0.5, 0.75, 0.9, 1, 1.25, 1.5, 2, 2.5, 3, 3.5, 4, 4.5, 5, 5.5, 6, 6.5, 7).

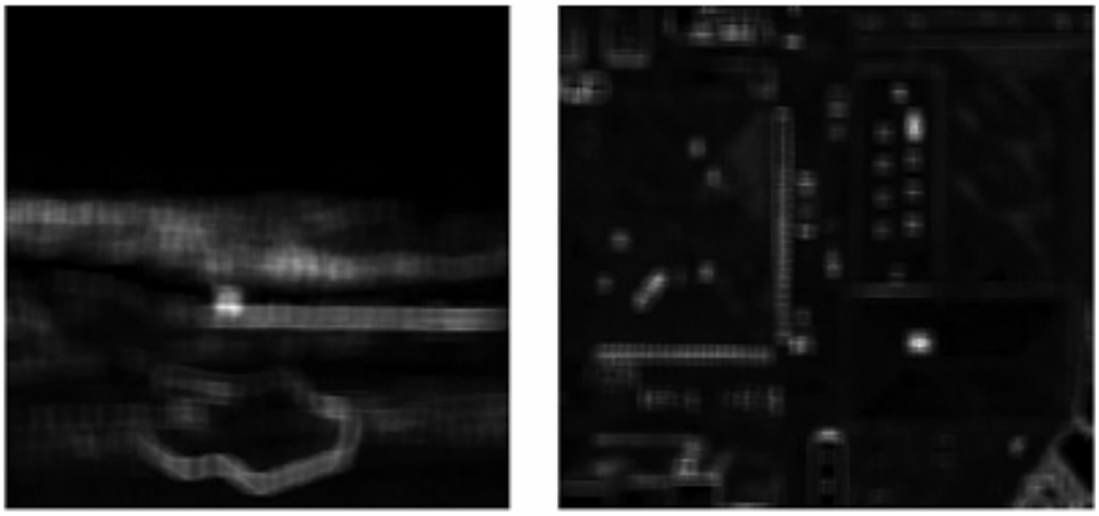

Figure 5.36: Response matrix of the scene: a vehicle (left) and a building (right)

Figures 5.37 through 5.40 show sample histograms of the cross-correlation function $c(m,n)$ obtained for $\theta = 0^{o}$ and 90^{o}. The analysis of the histograms allows one to identify a consistent pattern, which is significantly invariant to the value of the rotation angle θ. As the resolution level approaches its true value $SX = SY = 1$, the histogram exposes a greater degree of symmetry, with a noticeable peak value located close to the center of the histogram basis. As the resolution level changes toward smaller values, the histogram becomes more right-skewed, and its peak value moves toward the right end of the histogram basis. As the resolution level changes toward larger values, the histogram becomes flatter, and its center moves toward the right end of the histogram basis. This pattern allows one to crudely estimate the lower and upper bounds of the scaling factors. Thus, for the image of a vehicle, the lower and upper bounds of *SX* and *SY* are 0.25 and 4, respectively (see Figures 5.37 and 5.38). For the image of a building, the lower and upper bounds of *SX* and *SY* are 0.33 and 5.5, respectively (see Figures 5.39 and 5.40).

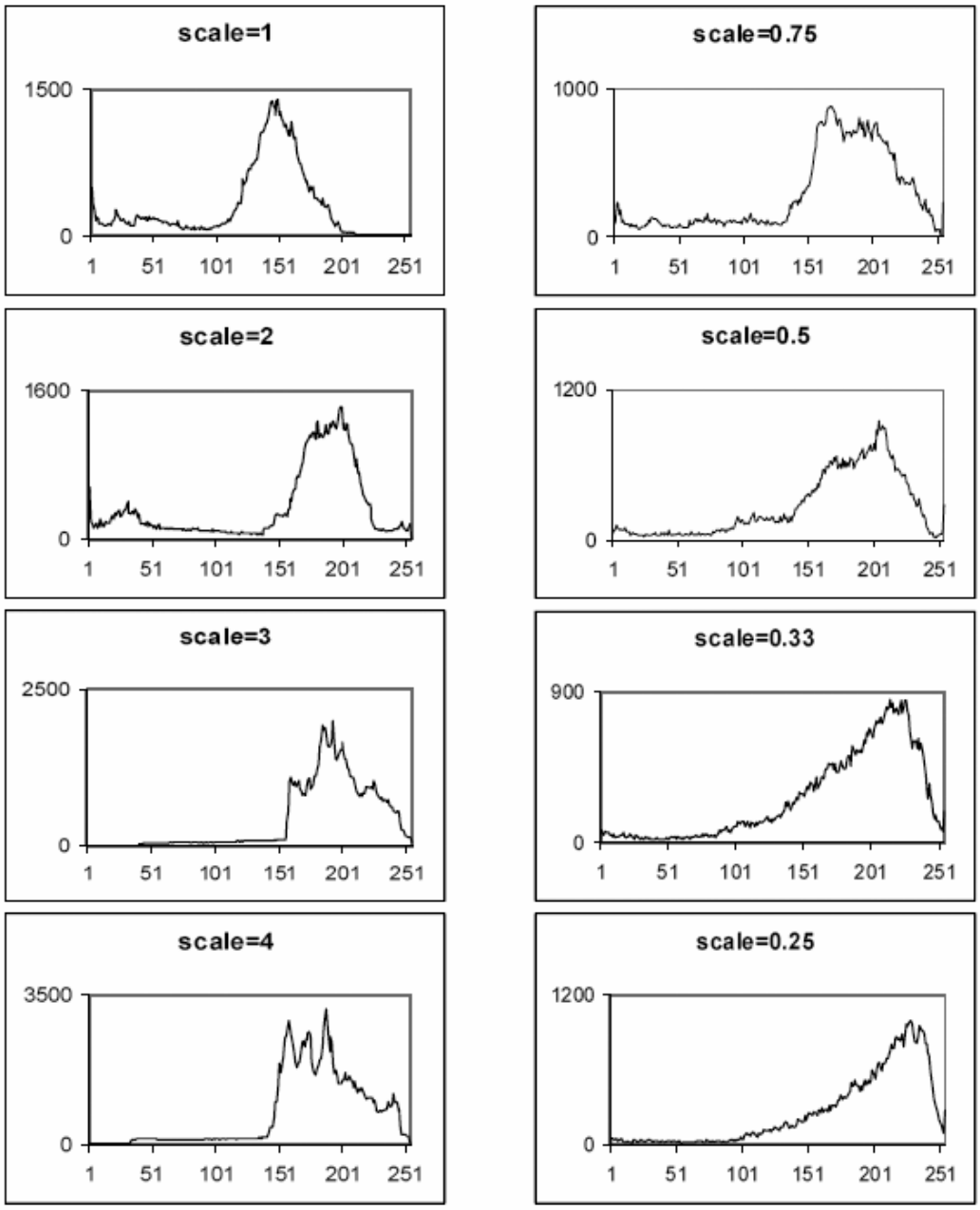

Figure 5.37: Histograms of cross-correlation of the vehicle responses, for $\theta = 0^{o}$

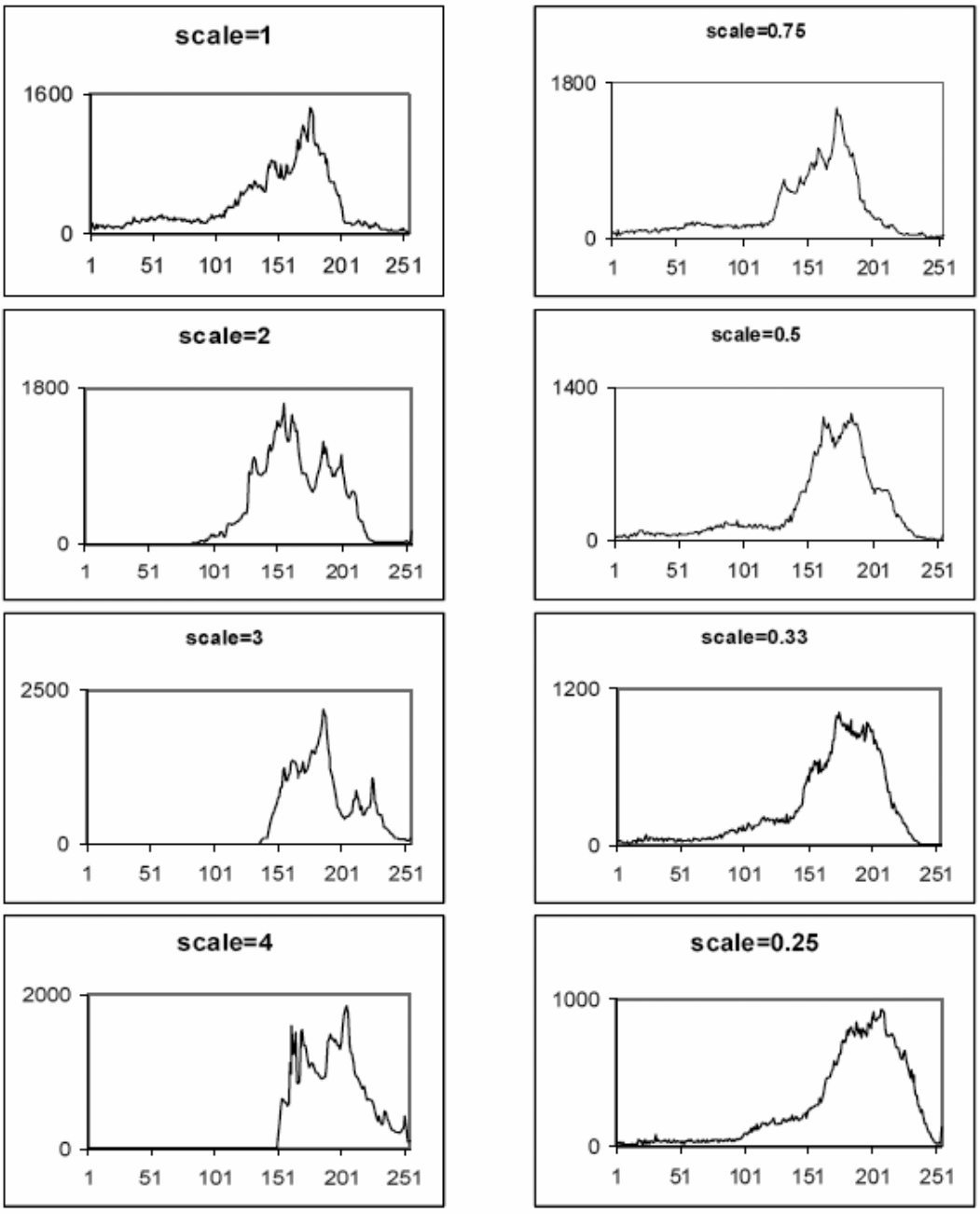

Figure 5.38: Histograms of cross-correlation of the vehicle responses, for $\theta = 90^{o}$

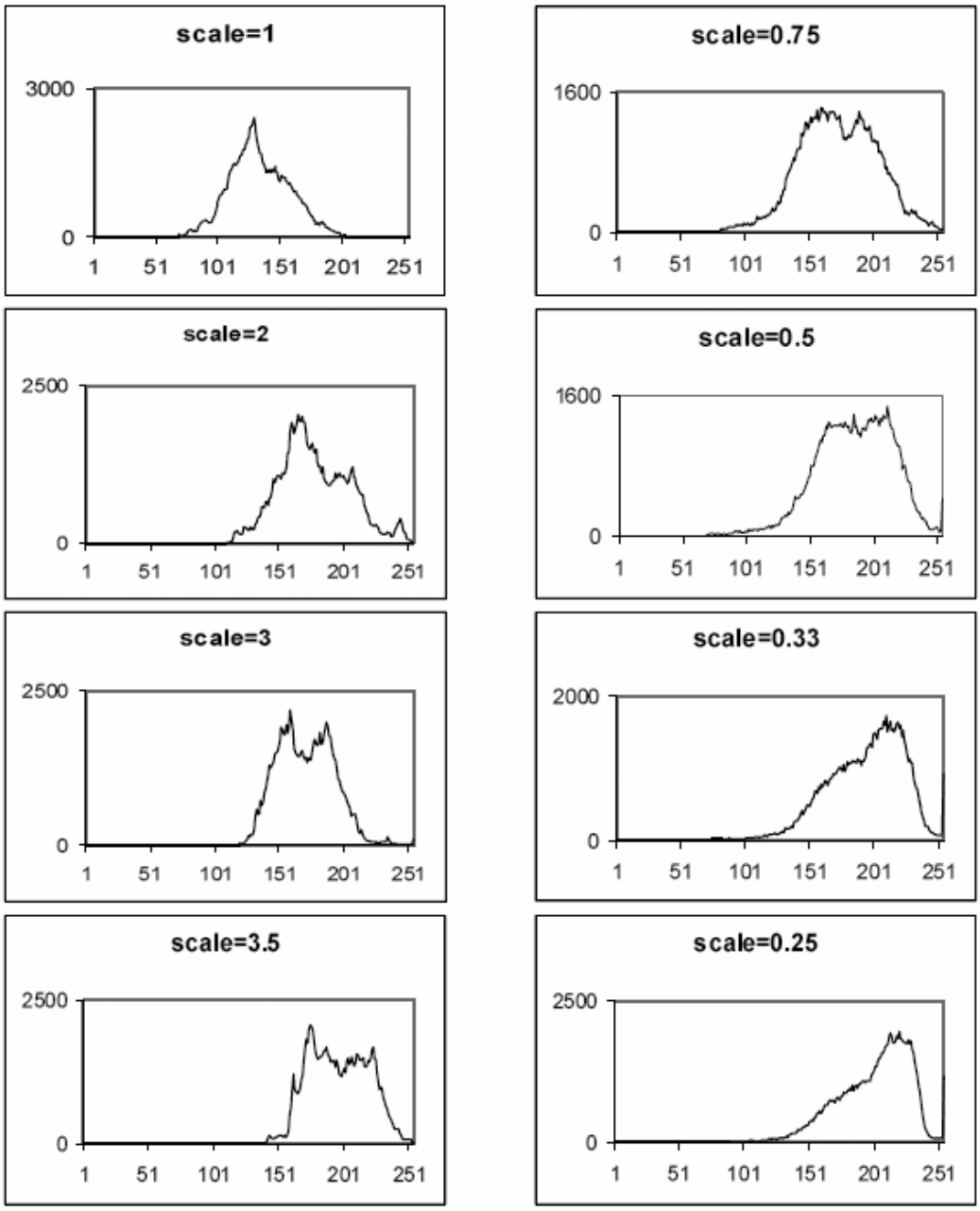

Figure 5.39: Histograms of cross-correlation of the building responses, for $\theta = 0^{o}$

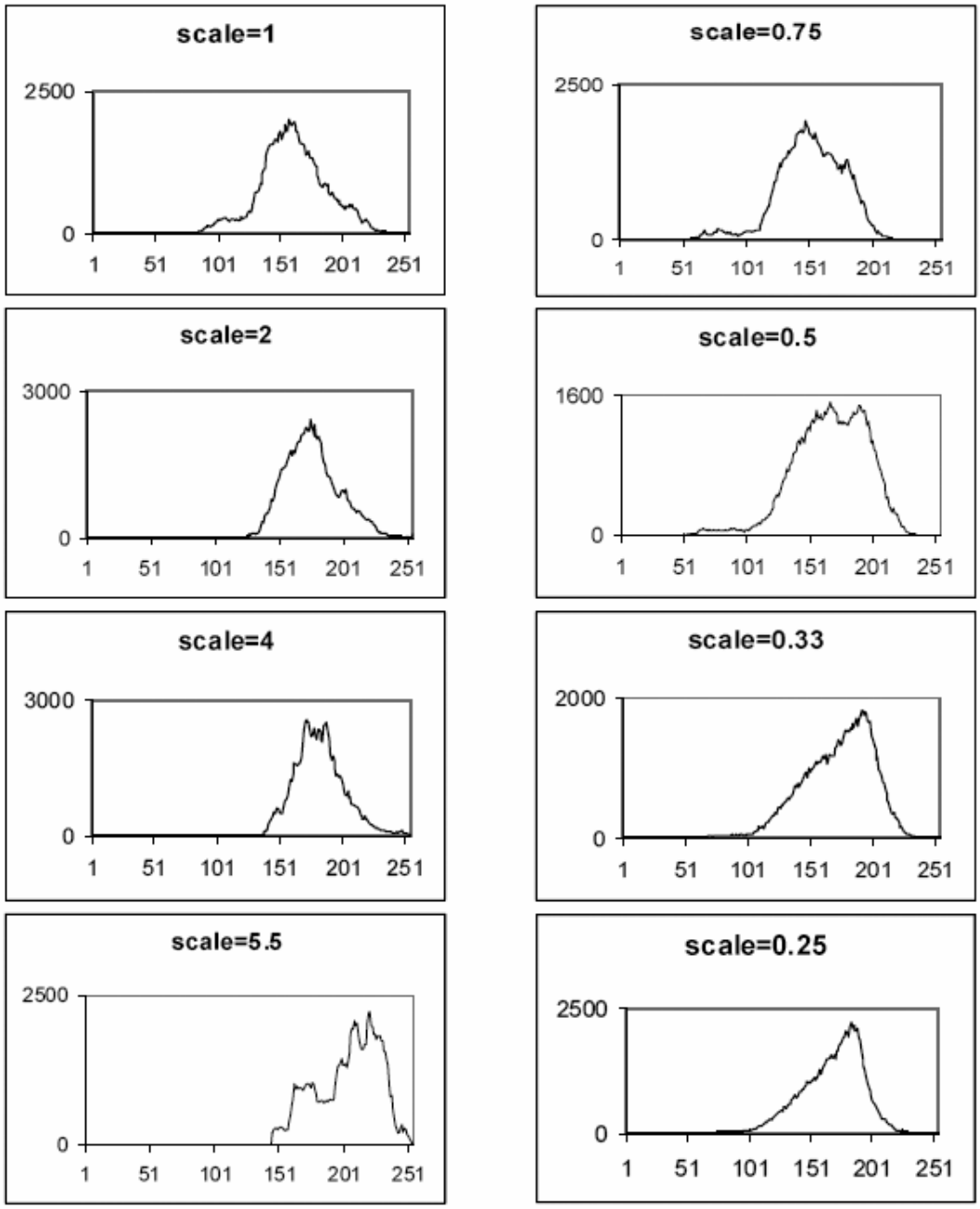

Figure 5.40: Histograms of cross-correlation of the building responses, for $\theta = 90^{o}$

5.8. Image Local Response in Multi-Sensor Fusion

An important part of information fusion, multi-sensor fusion often serves as the basis for automatic object and target recognition. The recognition task can be re-formulated as a

nonlinear global optimization problem, i.e., the search for a proper transformation that provides the best mapping between the images of the targeted object and the scene. Multi-sensor fusion maps images of the same scene received from different sensors into a common reference system.

Evolutionary algorithms have been successfully used for solving a few imaging problems related to object recognition. From the standpoint of the method used to evaluate the quality of the candidate solution, current applications of EAs in imaging applications can be broken into two main groups:

1. The first group of methods directly compares distributions of image pixel values. If the light conditions are changing between the images, the comparison becomes difficult, since no matching pixels can be found. Moreover, the task of comparison of different types of imagery, e.g., infrared and real visual images obtained from different types of sensors (i.e., in multi-sensor fusion), becomes significantly harder with these methods.
2. Methods in the second group attempt to find a set of salient characteristics, i.e., features that are common for the compared images. Choosing the appropriate features is not a trivial task, which becomes even more complex when images are misaligned and distorted by some kind of geometric transformation, e.g., affine or perspective.

Image local response has two important properties that can be effectively utilized in multi-sensor fusion:

1. Response emphasizes segments of an image that are most responsive to a particular geometric transformation. These segments identify the salient features of the image that are essentially important in solving image mapping problems.
2. Image local response can be viewed as an image transform $R(\boldsymbol{V})$ that maps the image onto itself, with the small perturbations of the parameter vector $\boldsymbol{V}$. Since $R(\boldsymbol{V})$ maps the image onto itself, the result of the mapping is largely invariant to the type of the sensor that is used to obtain the image.

The key idea of the proposed algorithm of object recognition from multiple sensors is to utilize the properties of Image local response, and to transform the originally stated problem of object recognition in real visual space G into a new problem of object recognition in image response space R. The latter is defined as a search space where the original image pixel values are transformed into the corresponding values of Image local response. The new problem can be formulated now as the search for an appropriate transformation A providing the correct aligning of responses $r_T(x,y)$ of the targeted object with the responses $r_S(x,y)$ of the scene.

The response matrices for both images are computed at the pre-processing stage. As in the original statement of object recognition problem, an individual chromosome encodes a trial vector of parameters $\boldsymbol{V}$ defining the sought geometric transformation A. The chromosome $\boldsymbol{V}^*$ that minimizes a measure of the difference F between the responses r_T and r_S of the images Img_T and Img_S identifies and aligns the recognized object. The sum

of the squared differences of the response values of the images Img_T and Img_S is used as a measure of the difference F between the images, as follows:

$$F = \frac{\sum_{\Omega} (r_T(x', y') - r_S(x, y))^2}{\Omega^2}, \quad (5.21)$$

where $r_T(x',y')$ and $r_S(x, y)$ are the response values of the images Img_T and Img_S, respectively, and Ω is the area of their overlap. The use of the imagery-invariant response values $r_T(x',y')$ and $r_S(x, y)$ in Formula (5.21) makes the proposed method of object recognition fundamentally different from the traditional methods based on the comparison of the actual pixel values $g_T(x',y')$ and $g_S(x, y)$. The algorithm using image response in multi-sensor fusion can be now formulated as follows.

Step 1: The matrix M_R of Image local response is computed during the pre-processing stage, for every participating image of the targeted object.

Step 2: The weighted matrix M_A of Image local response is formed, as follows:

$$M_A = \frac{\sum_{i=1}^{I} T_i M_{Ri}}{I}, \quad (5.22)$$

where I is the number of the object images, and T_i is the characteristic matrix particular for the sensor type used to obtain the image Img_i.

Step 3: During the evolutionary search, the weighted matrix M_A is compared against the response matrix M_S of the scene, in order to recognize the targeted object.

Figure 5.41: Synthetic images of the targeted object: a wireframe model (top, left), a principal model (top, right), and a textured 3-D model (bottom)

In order to illustrate the use of Image local response in multi-sensor fusion with HEA, computational experiments are conducted on 2-D synthetic grayscale images. The effect of different sensor types is simulated using the set of different images of the 3-D model of the targeted object [72], as shown in Figure 5.41. The set includes the wireframe, principal, and final textured models of the object, which exhibit significant difference in imagery type. A synthetic 300×300-pixel reference image Img_0 of the scene with the object is shown in Figure 5.42. The object is subject to a partial affine transformation A defined by the 3-dimensional vector $\boldsymbol{V} = \{DX, DY, SX = SY\}$, as well as to a general affine transformation defined by the 7-dimensional vector $\boldsymbol{V} = \{DX, DY, \Theta, SX, SY, SHX, SHY\}$, where DX and DY are the translations along the x and y axes; Θ is the rotation angle in the xy plane; SX and SY are the scaling factors along the x and y axes; and SHX and SHY are the shear factors along the x and y axes.

Figure 5.42: Synthetic scene with the targeted object

Parameter	***DX***	***DY***	***Θ***	***SX***	***SY***	***SHX***	***SHY***
Min value	0	0	0.0	1.0	1.0	0.0	0.0
Max value	299 255	299 255	6.28	4.0 3.0	4.0 3.0	2.0	2.0
Step size	1 2	1 2	0.1	0.1	0.1	0.1	0.1

Table 5.7: Ranges and step sizes of the parameters used in computational experiments

The object recognition problem is stated as a search for the vector of parameters $\boldsymbol{V}^*$ minimizing the difference F between the images Img_T and Img_S of the object and the scene, respectively. The search is conducted with the hybrid EA model. The ranges and step sizes of the sought parameters are presented in Table 5.7. The experiments are conducted in the real visual space G, i.e., by comparing the pixel values $g(x,y)$ of the original images, and in the response space R, i.e., by comparing the response values $r(x,y)$ of the images. The experiments have the following objectives:

1. To show that different types of imagery cannot be used for object recognition with EAs in real visual space *G*, because fitness functions corresponding to different imagery types, here the wireframe and solid models, have different optimum values in *G*.
2. To validate the use of image response space *R* in object recognition with EAs, by showing that fitness functions corresponding to the same imagery type, here for the solid model, have the same optimum values in both, the real visual *G* and response *R* spaces.
3. To show that different types of imagery can be used for object recognition with EAs in image response space *R*, because fitness functions corresponding to different imagery types, here the wireframe and solid models, have similar optimum values in *R*.
4. To show that the performance of the local DSM search does not degrade when the search is relocated into image response space.
5. To show that object recognition problem for misaligned and geometrically distorted images; and different imagery types can be successfully solved with HEA in image response space *R*.

Response matrices M_R are computed for all test images shown in Figure 5.41. The response matrices, as well as a sample surface chart of the response for the textured model, are presented in Figure 5.43. The responses of the wireframe and solid models exhibit a high degree of similarity; both responses highlight the main shape features of the targeted object. The wireframe response has more low-intensity details, which can be

discarded after applying an intensity threshold. The weighted response matrix M_A of the targeted object and the response matrix of the scene are shown in Figure 5.44. For simplicity, the identity matrix is used as the characteristic matrix T in Formula (5.22).

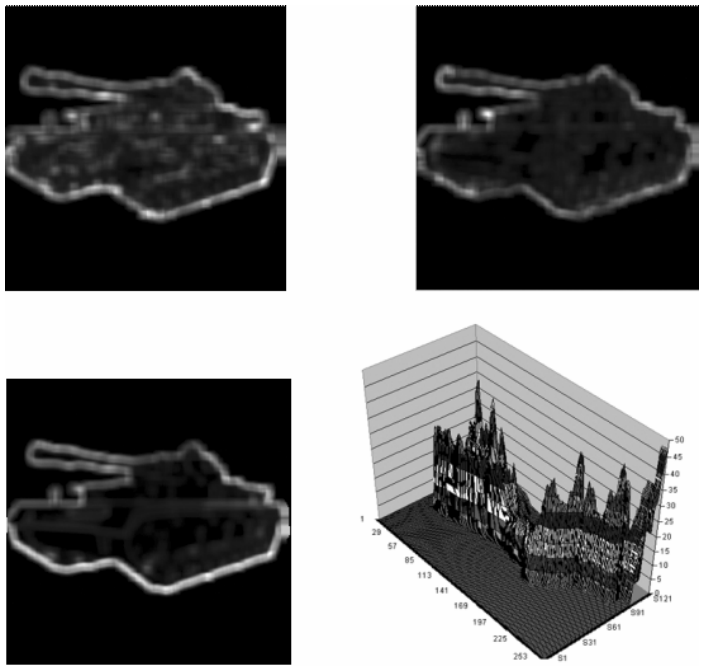

Figure 5.43: Image responses of the wireframe (top, left), the principal (top, right), and the textured 3-D models (bottom, left); a sample 3-D response of the textured 3-D model (bottom, right)

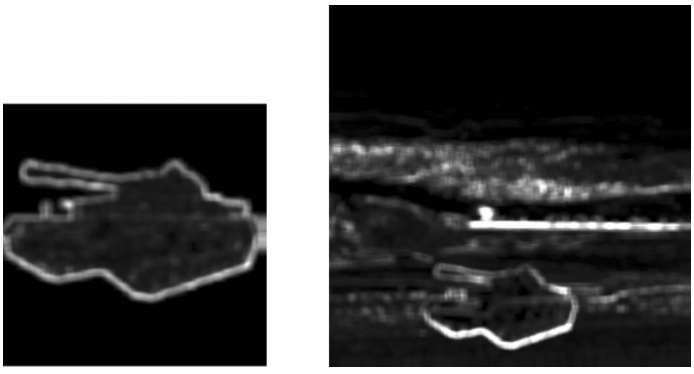

Figure 5.44: Averaged image responses of the object (left) and the scene (right)

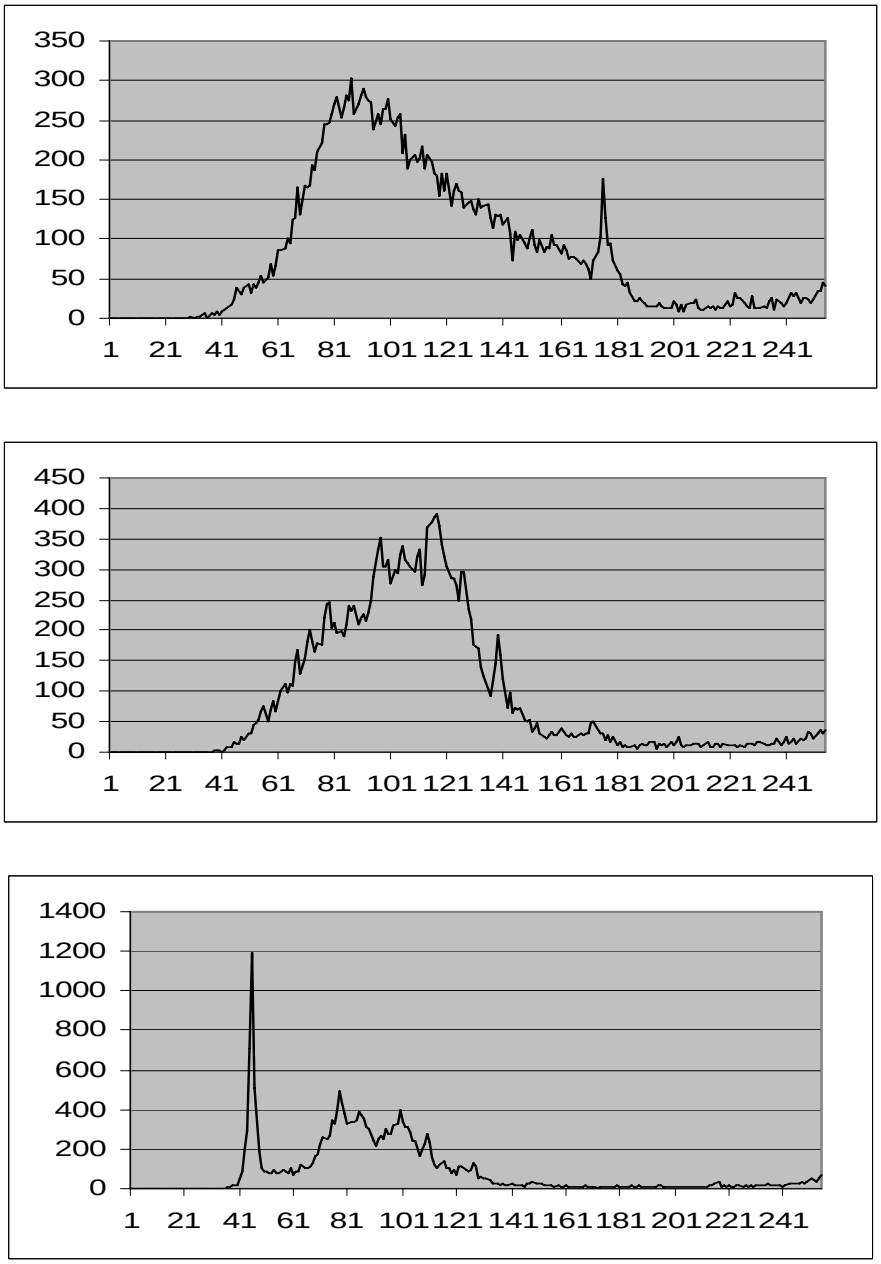

Figure 5.45: Image histograms (top to bottom): the wireframe, the principal, and the full textured 3-D models

The feasibility of conducting the search in real visual space *G* for different imagery types is assessed by comparing the gray values of the wireframe and solid models with the gray values of the scene. The histograms of the object images shown in Figure 5.45 display a

significant imagery difference between the wireframe, principal, and textured models, which makes the comparison of the actual image pixels in visual space a difficult, if not impossible task.

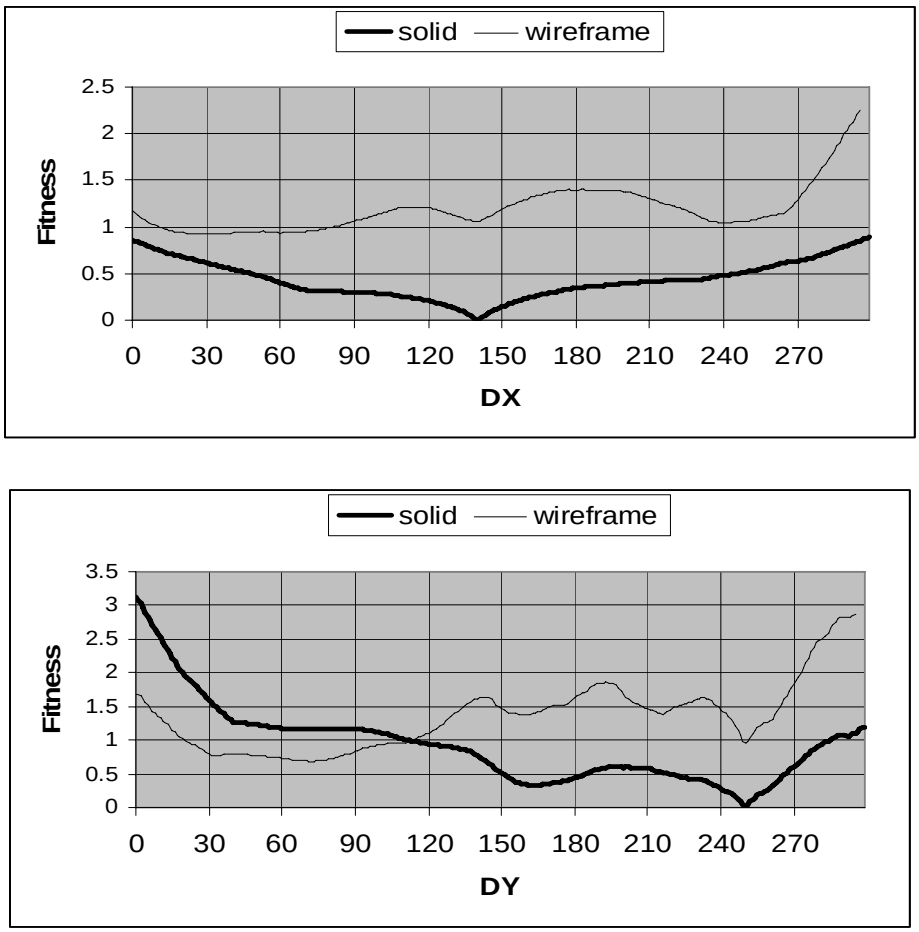

Figure 5.46: Fitness functions of the solid and wireframe models along the optimum cross-sections of the scene in real visual space *G*: *DY* = 250 (top), *DX* = 140 (bottom)

To compute the fitness function, the parameters Θ, *SX*, *SY*, *SHX*, and *SHY* of the chromosome ***V*** are set to their optimum values, while the translations *DX* and *DY* vary between their minimum 0 and maximum 299 values. The computed values of the fitness

function along the optimum cross-sections $DX = 140$ and $DY = 250$ for the both models are shown in Figure 5.46. The solid model clearly expresses global minima along both optimum cross-sections, which corresponds to the correct aligning of the object and the scene. The wireframe model exhibits only local minima at the correct locations along both cross-sections; it identifies the erroneous positions of the global minima, $DX = 32$ and $DY = 76$. This result clearly indicates that the wireframe model cannot be used for recognizing the realistic solid object in real visual space G, because of the significant difference in the imagery type between the two models. The use of the actual images of the wireframe or principal models results in the faulty mapping between the template and reference images.

As opposed to the actual gray values of the models, the response matrices M_R computed for the wireframe, principal, and textured models display a higher degree of similarity, which can be seen on the histograms computed for the images of M_R - see Figure 5.47. The values of the fitness function along the optimum cross-sections of the scene are computed for the solid model in image response space R using Formula (5.21), i.e., by comparing the response values $r(x,y)$ of the solid model and the scene. The computed values plotted in Figure 5.48 clearly indicate the correct locations of the global optimum in the both, the x and y directions, i.e., the optimum parameter values are $DX = 140$ and $DY = 250$. The fact that fitness function corresponding to the solid model does not change its behavior when the search is conducted in image response space, validates the feasibility of using response space R, instead of real visual space G, in object recognition with EAs.

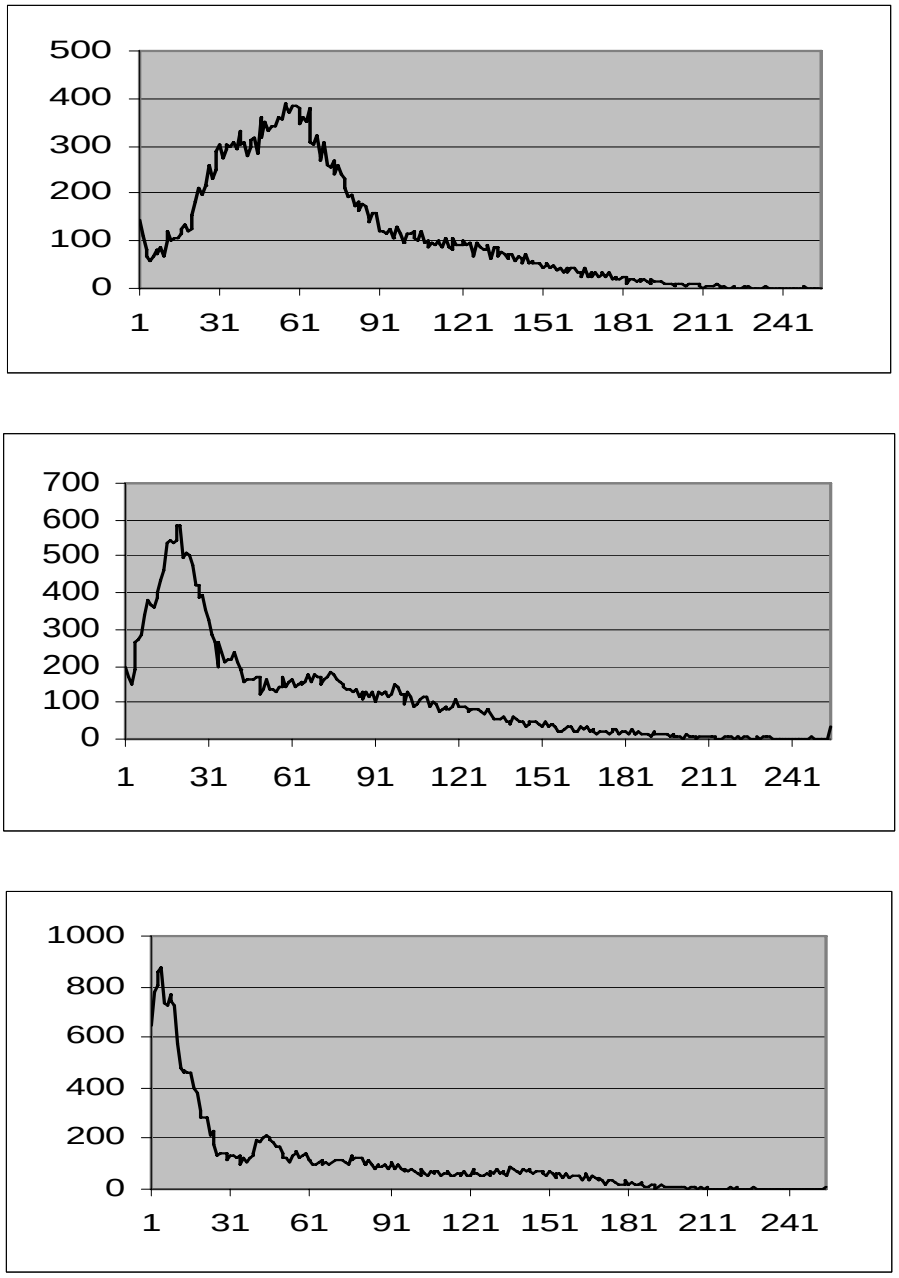

Figure 5.47: Histograms of the response matrices (top to bottom): the wireframe, the principal, and the full textured 3-D models

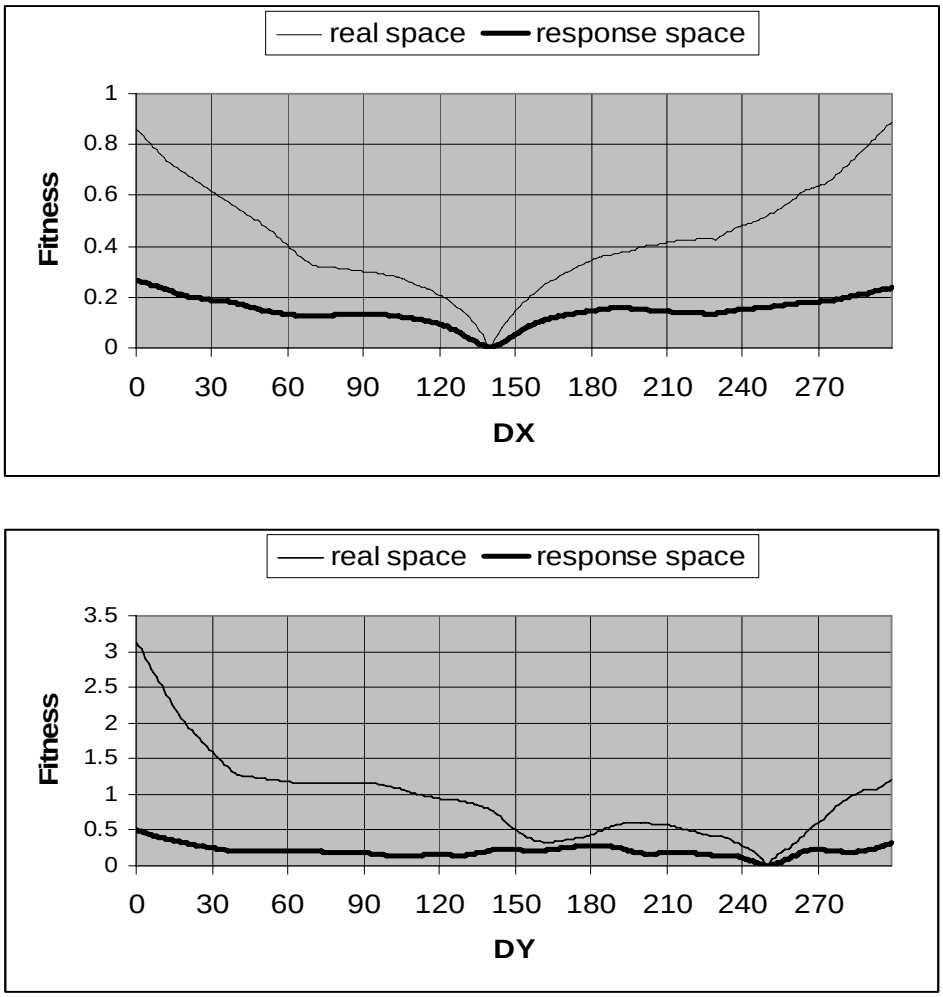

Figure 5.48: Fitness functions of the solid model along the optimum cross-sections of the scene in the real *G*, and in the response *R* spaces: *DY* = 250 (top), *DX* = 140 (bottom)

Once the validity of relocating the search space from real visual space *G* into response space *R* has been proven, the most important question arises, whether the behavior of fitness function *F* in response space is similar for different imagery types, here for the wireframe and solid models. The computed values of the fitness function along the optimum cross-sections for the both models are compared in Figure 5.49. The global minima along the *x* and *y* directions computed for the wireframe model correspond to *DX*

= 144 and *DY* = 254, which are very close (with the maximum error 3%) to the minimum positions for the solid model. This important result clearly indicates that the wireframe model can be used for recognizing the realistic solid object in image response space *R*.

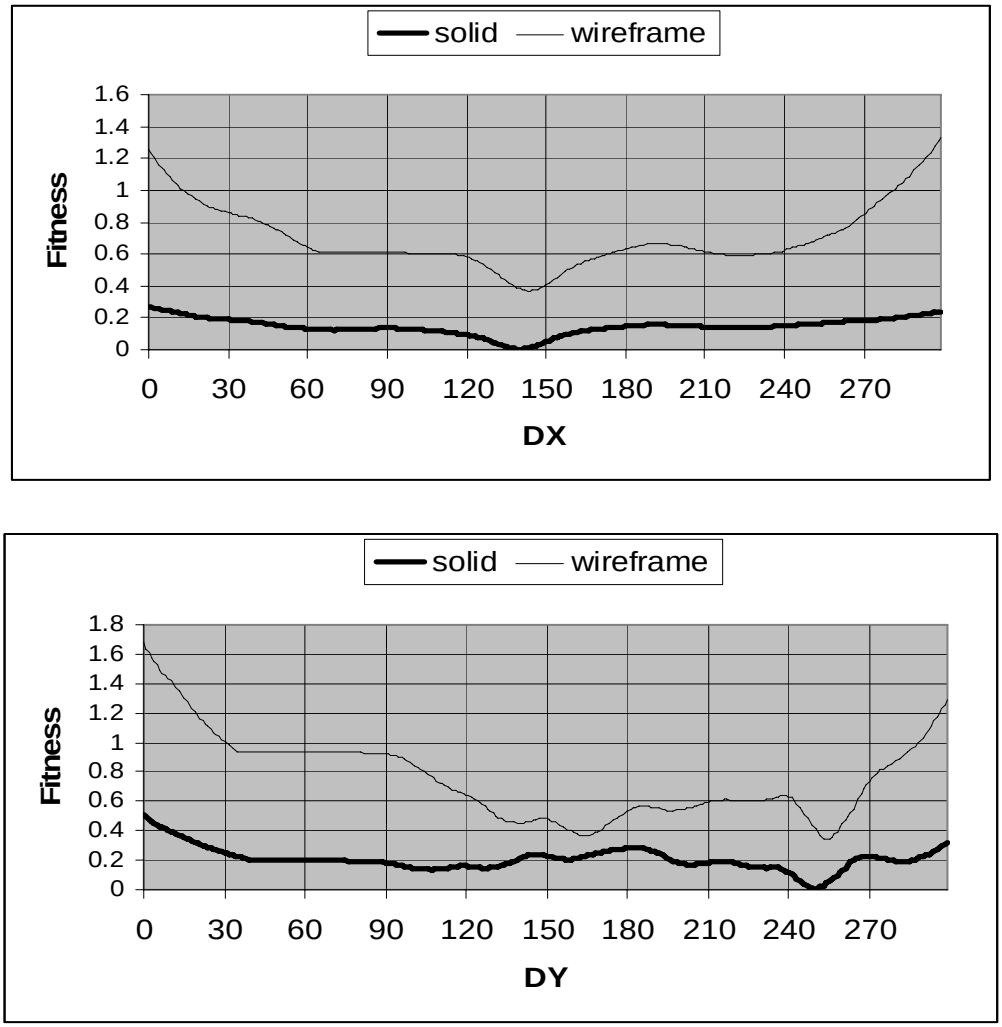

Figure 5.49: Fitness functions of the solid and wireframe models along the optimum cross-sections of the scene in the response space: *DY* = 250 (top), *DX* = 140 (bottom)

One of the main features of HEA is the utilization of local search within the global evolutionary procedure. The objective of the next experiment is to show that the

performance of the local DSM search does not degrade when the search is conducted in the response space *R*. The performance is assessed as the number of fitness evaluations required to find the optimum solution. A series of 100 test runs of the local DSM search for the solid and wireframe models in the real *G* and response *R* spaces, are performed. For every test run, the initial simplex is placed in the proximity of the optimum solution using the following technique. The value of each parameter of the vector ***V*** for each of the vertices is independently drawn at random, with the uniform probability, from the (±10%) range of the corresponding parameter domain centered at the parameter's optimum value. For example, the value of the translation *DX* for a 300×300-pixel image of the scene is drawn from the interval (140.0 ± 30). Figure 50 shows a sample image of the transformed solid model.

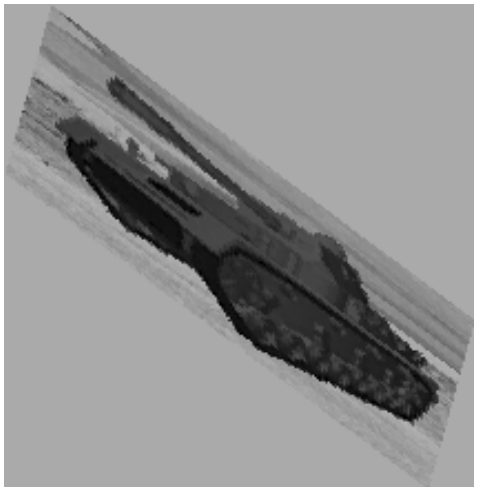

Figure 5.50: Sample transformed image of the solid model obtained with the 10% parameter range about the optimum solution

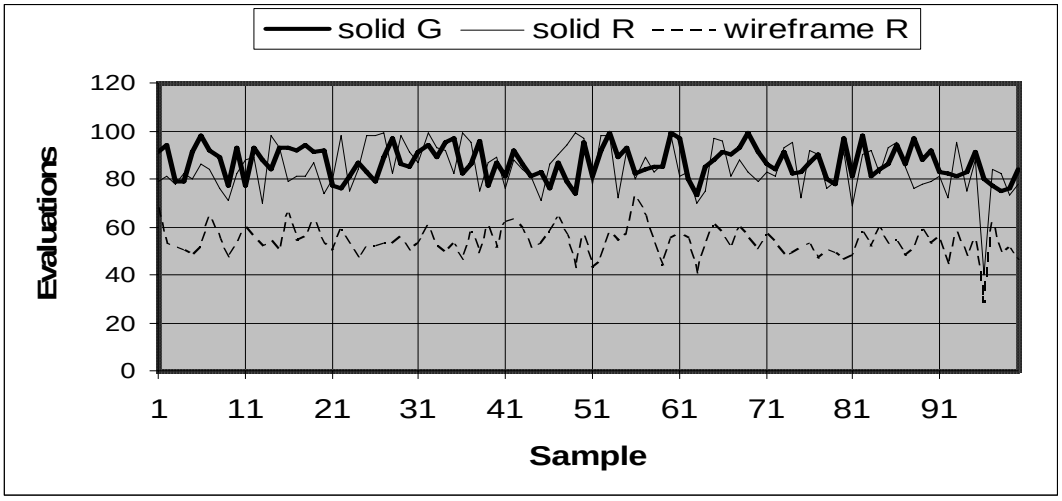

Figure 5.51: Comparative performance of the local DSM search for the solid and wireframe models in the real *G*, and in the response *R* spaces

Comparative results for the number of fitness evaluations for all test runs are shown in Figure 5.51. The performance of the DSM search for the solid model is similar in both, the real and response spaces: the total number of fitness evaluations over 100 runs is 8681 in the real space, and 8472 in the response space. The performance of the local DSM search for the wireframe model in the response space is even better than the performance of the solid model: the total number of fitness evaluations is 5354. These results assure that the performance of the local DSM search, at least, does not degrade, when the search is conducted in the response space, for the different imagery types.

Finally, the hybrid model of Evolutionary algorithm is used to solve the object recognition problem for the different imagery types. The distorted response image of the wireframe model is used to recognize the targeted object in the response image of the scene containing the solid model. In the first test run of the algorithm, the wireframe

model is subject to an affine transformation defined by the 4-dimensional vector $\boldsymbol{V} = \{DX, DY, SX, SY\}$, where $SX = SY$ (i.e., for the isotropic scaling of the image). In the second test run, the wireframe model is subject to an affine transformation defined by the 5-dimensional vector $\boldsymbol{V} = \{DX, DY, \Theta, SX, SY\}$, where $SY = 2SX$ (i.e., for the non-isotropic scaling of the image). The first test run is performed with the population of 137 chromosomes initially selected at random with the uniform probability. The population size in the second test run is 260. The images of the object used in the search, and the results of the recognition are shown in Figure 5.52, where the rectangular segment of the scene corresponding to the image of the targeted object is highlighted. The algorithm is able to correctly recognize and align the object in the scene, in the both test runs.

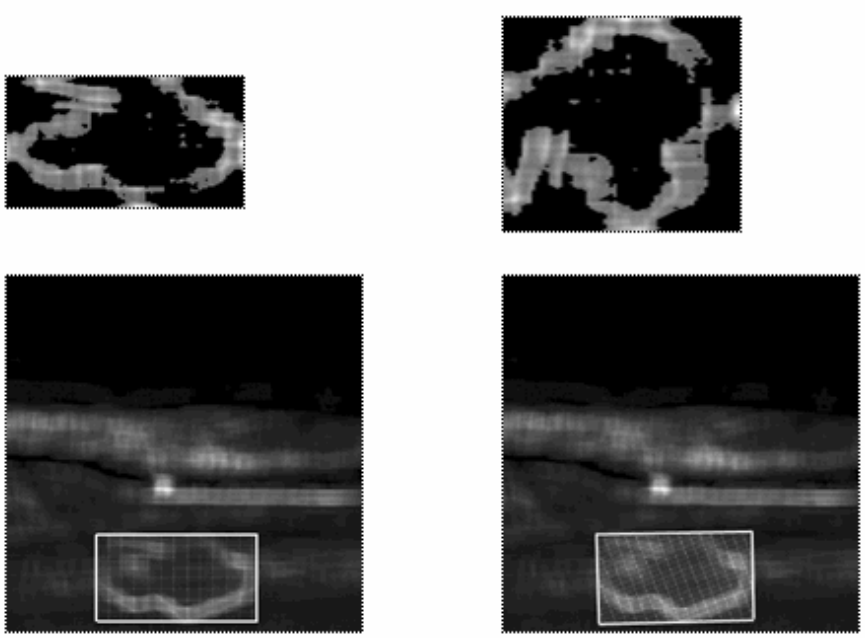

Figure 5.52: Images of the targeted objects (top row) and the recognition results (bottom row) for the 4-parameter test (left), and for the 5-parameter test (right)

10% shift	**Parameter values**					**Fitness**
	DX	***DY***	***θ***	***SX***	***SY***	
None	78.3	215.8	0.0	2.0	2.0	0.0563
DX	70	216	0.0	2.0	2.0	0.1477
DY	78	194	0.0	2.0	2.0	0.1497
θ	78	216	0.63	2.0	2.0	0.2265
SX	78	216	0.0	2.2	2.0	0.1325
SY	78	216	0.0	2.0	2.2	0.1568
SX, SY	78	216	0.0	2.2	2.2	0.1806

Table 5.8: Parameters for the optimum solution, and for the neighboring points shifted by 10% from the optimum point

Table 5.8 gives the comparison of the parameters and fitness values for the optimum solution and the neighboring points in the parameter space. Each neighbor has one of the components $\boldsymbol{V} = \{DX, DY, \theta, SX, SY\}$ shifted by 10% from its optimum value. In addition, the fitness values of nine solutions picked at random outside the 10% interval are computed, with the average fitness value $F = 0.101$. The large difference in the F values, where the minimum value corresponds to the optimum solution, provides the experimental support for the use of the response space in solving the object recognition problem as an optimization problem. The total number of fitness evaluations in the first test is 119454, with 12153 evaluations attributed to the local DSM search. The total number of fitness evaluations in the second test is 117982, with 13898 evaluations attributed to the local DSM search.

Computational experiments conducted on a test set of 2-D synthetic grayscale images allow one to draw the following conclusions:

1. Different types of imagery cannot be directly used in the real visual space for object recognition with HEA, because fitness functions corresponding to the images of different types have different optimum values in this space.
2. Evolutionary search can be relocated into image response space, because fitness functions corresponding to the real image and its response counterpart have similar optimum values in both, real and response spaces.
3. Different types of imagery can be used in image response space, because fitness functions corresponding to images of different types have similar optimum values in the response space.
4. The performance of the local DSM search does not degrade when the search is relocated into image response space.
5. Object recognition problem for misaligned and geometrically distorted images; and different imagery types, can be successfully solved with HEA in image response space.

5.9. Summary

This Chapter introduces a particular image transformation called Image local response. The latter is defined as the variation of the objective (i.e., fitness) function occurred because of a small variation of the parameter vector. Response provides the algorithm with a fairly general method that can autonomously obtain essential information about an image, and has the following important properties that make it a valuable tool in solving imaging optimization problem with Evolutionary algorithms:

1. Response senses the distribution of pixel values across the image, and thus provides a crude estimation of the behavior of fitness function.
2. Response emphasizes the areas of the image that are most responsive to the applied geometric transformation.
3. Since response is computed by mapping image onto itself, it is not particularly sensitive to the imagery type.

The algorithms and the computational experiments presented in this Chapter demonstrate that the above listed properties of Image local response can be successfully used to improve the performance of the hybrid EA model, in the following main directions.

Image response accentuates those segments of the image that change the most during its transformation. Therefore, response extracts an important feature of the image, its dynamic contents, which can be used for image reduction and selective fitness evaluation, thus reducing the effective number of evaluations. Computational experiments are conducted with two image sets, in the case of the 5-dimensional mapping problem, with the full image, and with the selective fitness evaluation based on a binary mask obtained with Image local response. Full images require $E = 9119$ and $E = 5236$ evaluations, and give the minimum fitness values $F = 0.00876$ and $F = 0.00317$, for the first and second sets, respectively. The use of selective fitness evaluation requires only $E = 6088$ and $E = 2495$ evaluations, giving similar minimum fitness values $F = 0.01154$ and $F = 0.00248$, for the first and second sets, respectively.

Since image response provides a model approximating the behavior of fitness function in a small locality, it can be used to devise a mechanism that controls the movement of a DSM simplex and makes the latter adaptive to the local properties of fitness function. This adaptive mechanism allows one to reduce the overall computational cost of fitness evaluations associated with the local DSM search. Experiments conducted on image sets under the 5-dimensional mapping show the reduction of the number of fitness evaluations between 27.2% and 44.9%, without the loss of the quality of the optimum solution. The reduction of the computational cost of the DSM fitness evaluation with image response comes at a cost of large number of evaluations of local responses at different points of the image. The amount of computations can be reduced by pre-computing the response surface of the entire image; engaging a self-organizing network, in order to classify image areas according to their responses; and building the response map of the image. Experiments on the test sets, in the case of the 5-dimensional mapping, show that the use the response map results in the reduction of the number of response evaluations between 58% and 70%.

During the operations of crossover and selection, Image local response can be used to limit the creation of new candidate solutions to only those sub-areas of the search space that might most likely contain the optimum solution. The operation of cross correlation of image response matrices is performed and scaled to the range of (0,1). The resulting correlation matrix serves as the likelihood matrix applied during selection and crossover, to highlight the sub-areas in the reference image onto which the template image most likely can be mapped. The use of this technique allows one to find the correct 5-

dimensional mapping on a test set after 24 generations, when the regular algorithm stalls in a local minimum point. When applied to the second test set, this technique allows one to reduce the computational cost of the mapping to 7 generations, as opposed to 23 generations spent by the regular algorithm.

Cross correlation of image response matrices can be used in the case of multi-resolution image mapping. On each resolution level, the cross correlation of the response matrices of the reference and template images outlines the potential locations of the latter. Once the locations are identified, the algorithm zooms in on the found locations. The zooming process continues until the final resolution level is reached, at which point HEA can be utilized to find the correct mapping. The same approach can be used to find the lower and upper bounds of the scaling factor in the parameter vector that defines the transformation between the images. The response correlation technique is illustrated on sample image sets.

Image local response can be viewed as image transform that maps the image onto itself making the result of the mapping largely invariant to the imagery type. This property can be efficiently used to solve imaging optimization problems arising in multi-sensor fusion, when images subject to the mapping are obtained from the different sensor types. Experiments conducted on a sample test set, in the cases of 4- and 5-dimensional mapping problems, show that conducting the search in the response space, rather than in the actual image space, allows one to successfully map images of different types, particularly, a wireframe object model onto the scene containing the solid object model.

Chapter 6: Multiple Populations and Multi-Objective Optimization of a Piece-Wise Affine Transformation

6.1. Introduction

Analysis of the means of hybridization of the evolutionary search with other computational methods given in Chapters 4 and 5 of this thesis, allows one to develop an advanced EA-based computational model which can be applied to solving difficult problems of imaging optimization. In particular, three sample problems are illustrated in this Chapter:

- human body registration;
- recognition of a 3-dimensional object in the 2-dimensional space;
- elastic registration of medical images.

All the above mentioned problems can be essentially stated as a search for a proper mapping between the images of a template and a scene, when the following conditions are present:

1. Two or more template images are used to represent different views of the same object.
2. The object of mapping undergoes significant distortion caused, e.g., by an arbitrary rotation in the 3-D space; such a mapping cannot be defined by a single transformation vector.

3. The difference between the images cannot be formulated as a single fitness function; consequently, the search has to deal with multiple objectives of the optimization.

In order to accommodate multiple template images, the advanced computational model uses multiple populations, where each template is represented by its own independent population. Since template objects can undergo significant distortion, they are broken down into *k* sections, such that each section can have its own transformation vector $\boldsymbol{V_k}$. This approach corresponds to a piece-wise approximation of the actual image transformation $A(\boldsymbol{V})$. The algorithm also assumes that the object in the template image has some prominent feature, in the form of a body, or a trunk, to which other parts of the object are attached. Such a feature will be called a hull, throughout this Chapter; the transformation of the hull can be defined by a single vector $\boldsymbol{V_A}$ of the general affine transformation, and by a complementary vector $\boldsymbol{V_D}$ of elastic deformations which describes the deviation of the actual hull transformation from the vector $\boldsymbol{V_A}$.

In its most general form, the entire algorithm works as two relatively independent passes, global search and local correction. Global search attempts to find the optimum solution for the hull transformation, while local correction attempts to find the optimum piece-wise approximation of the actual image transformation using the hull transformation as its initial approximation. Because of the complex structure of the template model and a two-phase search algorithm, one expression for fitness function is not sufficient. The

search is conducted in a multi-objective space using different expressions for fitness function at the different stages of the algorithm.

The advanced computational model presented in this Chapter is built upon the following concepts developed in Chapters 4 and 5:

1. Template objects and a scene are represented by their local responses, i.e., the entire search, from the very beginning, is conducted in image response space, rather than in the actual visual space.
2. Algorithm intensively uses local search, as a means of improving the quality of the better fraction of the population.
3. Algorithm keeps track of the usage of the entire parameter space, in order to provide all parameters with a fair opportunity to be represented in the population, and to mix different components, thus increasing the diversity of the population.

6. 2. Advanced Image Model Based on Image Local Response

As stated in Chapter 4 (section 4.6.4), the model of the image participating in the evolutionary search has to represent the most important image features. Image local response defined in Chapter 5 (section 5.2), provides such essential image representation. Chapter 5 (section 5.8) describes the usage of image response in multi-sensor fusion, where the initial optimization problem in the real imaging space is transformed into the optimization problem in the response space. To further develop this approach, the advanced model of an image participating in the evolutionary search is described in this section. The model is computed in accordance with the algorithm shown in Figure 6.1.

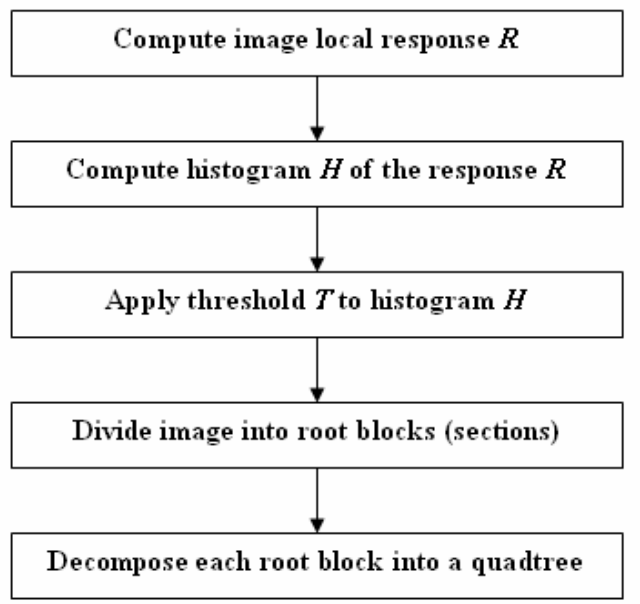

Figure 6.1: Algorithm for building the advanced image model based on Image response

In addition to the regular hierarchical quadtree structure [121], each section's quadtree is serialized, i.e., converted into a sequential string of nodes, which simplifies and speeds up all computational operations associated with the quadtree processing. Every node of the quadtree is represented by its mean response value which participates in the evaluation of the node fitness. To illustrate the process of building the model of an image, Figure 6.2 shows the sequence of transformations following the algorithm presented in Figure 6.1. The original image has dimensions 210×210-pixel and contains the total of 44100 pixels. The threshold applied to the histogram defines the accuracy of the model. In Figure 6.2, only 1% of the larger gray values are retained in the histogram, which makes the threshold equal to 45. The quadtree decomposition is based on a specified tree depth and the acceptable mean value of the difference in gray values of pixels comprising one block of the tree. For the image shown in Figure 6.2, the depth of the tree is set to 6, and the mean value of the difference between pixels of the same block is 1%. The final quadtree

model has 18 root blocks, or sections, and contains the total of 1314 nearly homogeneous nodes of the smallest size 2×2-pixel.

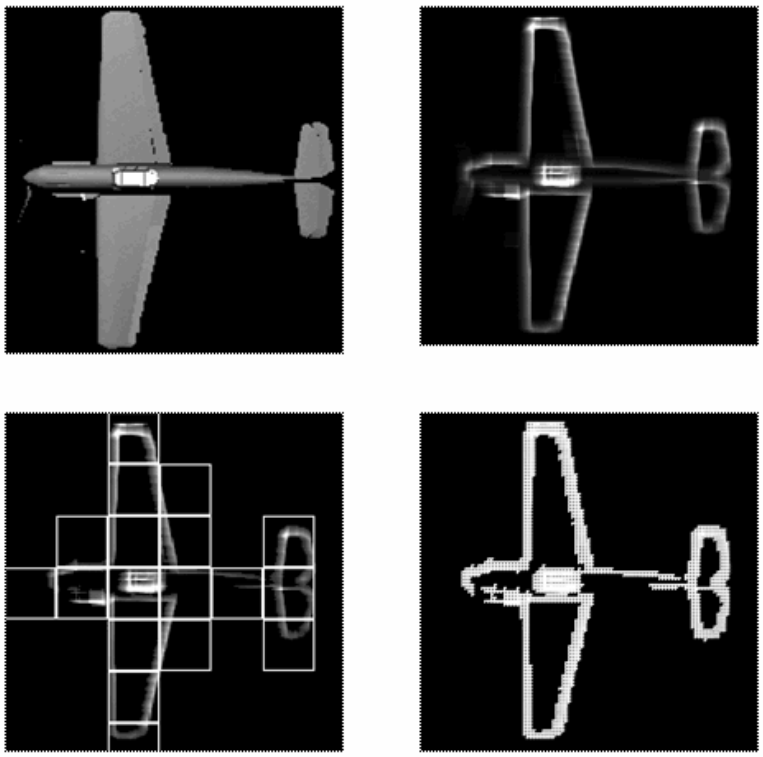

Figure 6.2: Sequence of transformations used to obtain the response model of the template image (top to bottom, left to right): the original image; image local response; response histogram with the outlined root blocks; the final image quadtree

Figure 6.3 shows a zoomed-in view of the same section of the image in its original response form, and after its transformation into a quadtree. As one can see, the nodes of the quadtree preserve fairly well the distribution of the original response values. The reduction in the number of operations during the fitness evaluation can be defined by the ratio of the total number of quadtree nodes to the total number of image pixels, i.e., 100×(1 - 1314 / 44100) = 97%.

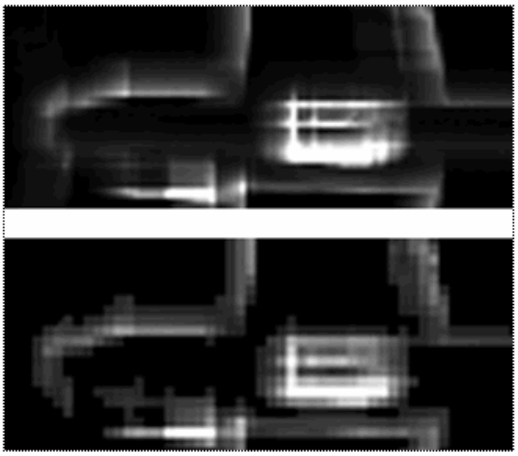

Figure 6.3: Original response image (top) and its quadtree transformation (bottom)

During the first phase of the evolutionary search, when the algorithm attempts to find the global location of the template image, the hull of the image is processed. The model of the hull includes the outline of the main part of the image represented as a double-linked list of nodes of a specified size and a set of internal black nodes of a specified size. The model can be built interactively, by simply drawing the outline of the hull. The hull has two important features participating in fitness evaluation:

- total length of the outer contour L_{HC} computed as a sum of all distances between the contour nodes;
- area covered by the hull A_H computed as the number of the black nodes enclosed inside the hull.

A zoomed-in sample model of the hull corresponding to the image shown in Figure 6.2 is presented in Figure 6.4. The outer contour of the hull is composed of the square nodes of the size of 2×2-pixel; the internal black nodes have the size of 1×1-pixel.

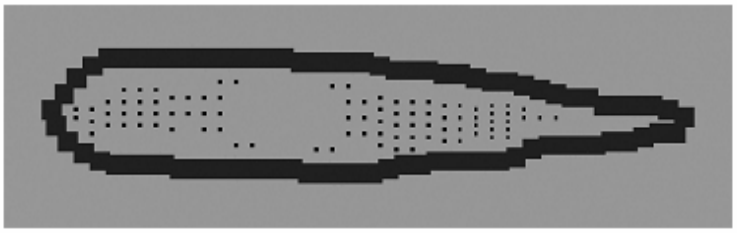

Figure 6.4: Sample model of the image hull

6.3. Global Search for the Optimum Hull Transformation

The first pass of the algorithm works with the image of the object hull, under the assumption that the full transformation T of the hull can be represented as a superposition of a general affine transformation A (a core transformation), and a complementary elastic piece-wise transformation (deformation) D, as follows:

$$T = A + D. \tag{6.1}$$

Parameters of the core affine transformation A are defined by the vector

$$\boldsymbol{V_A} = \{DX, DY, \theta, SX, SY, SHX, SHY\}, \tag{6.2}$$

where DX and DY are the translations along the x and y axes, θ is the rotation angle, SX and SY are the scaling factors along the x and y axes, and SHX and SHY are the shear factors along the x and y axes. Parameters DX, DY, and θ correspond to the rigid body transformation, whereas parameters SX, SY, SHX, and SHY define the global distortion of the image, when the appropriate factors are non-isotropic, i.e., when $SX \neq SY$ and $SHX \neq SHY$.

Parameters of the complementary elastic piece-wise transformation D are defined by the following vector:

$$\boldsymbol{V_D} = \{DX_{v1}, DY_{v1}, DX_{v2}, DY_{v2}, \ldots, DX_{vN}, DY_{vN}\}, \quad (6.3)$$

where $\{DX_{vn}, DY_{vn}\}$ is the partial deformation vector along the normal v to the n-section of the image template Img_B, and N is the total number of the template sections. While the vector $\boldsymbol{V_A}$ defines the core transformation of the hull, the vector $\boldsymbol{V_D}$ describes the deviation of the actual shape of the hull from its core transformation.

The overall strategy of the multi-population, multi-objective algorithm of the global search is presented in Figure 6.5. The algorithm starts with forming the initial population. Commonly used random generation of chromosomes is not well suited for the search that intensively uses local optimization procedure, since random chromosome placement will create many excessive and unnecessary individuals within the reach of the local search procedure. This consideration, together with the fact that random generation that does not support a fair participation of all parameter ranges in the search, which is discussed in Chapter 4 (section 4.4.2), in relation to the efficiency of mutation, leads to the need of expanding the mechanism of mutation with memory to the overall control over forming the population genotype. Such mechanism is implemented in the form of a frontal algorithm of memory management whose implementational details are discussed in section 6.3.1. The algorithm has the following important features:

1. It breaks the parameter space into regular cells whose size is limited by the capacity of the local search procedure.
2. It keeps track of the usage of each cell in crossover, mutation, and local search, and attempts to leverage this usage in such a way that all cells would have a fair representation in the population.

3. It attempts to mix parameter ranges in such a way that the population has a greater degree of diversity.

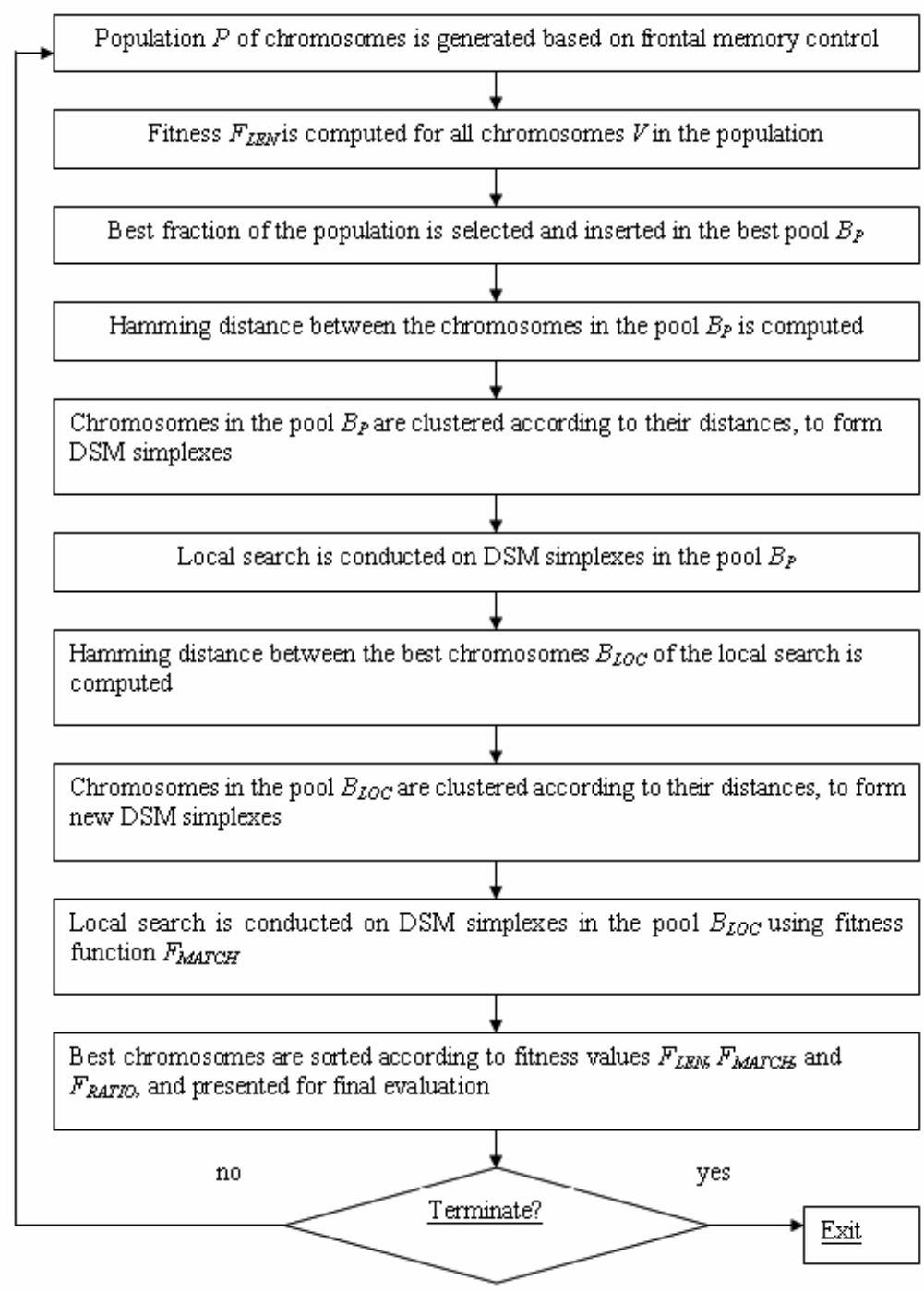

Figure 6.5: Algorithm of the global search for the elastic transformation of the object hull

When two or more template images are used in the search, multiple chromosome populations are generated by the algorithm, such that each template image has its own independent population. Since all populations attempt to map different templates onto the same reference image, the populations have to be synchronized, i.e., aligned with each other. The templates represent different views of the same objects which can generally have different values *DX, DY, SX,* and *SY,* of the core transformation defined by the vector $\boldsymbol{V_A}$. However, if one request that all templates are aligned on the rotation angle before the search begins, they have to retain their alignment throughout the search. This means that all populations have to be synchronized on the rotation angle when the populations are generated at the beginning of every global iteration. In fact, the rotation of the object in the 2-D space, in the case of the affine transformation, is defined by three components of the vector $\boldsymbol{V_A}$: rotation angle θ, shear factor *SHX*, and shear factor *SHY*. Therefore, all populations corresponding to the image templates have to be synchronized on the three above mentioned parameters.

Once the new chromosome population has been generated, fitness values are computed for all chromosomes in a batch manner. A multi-objective approach to fitness evaluation is undertaken. In the process of the global search, the following three objectives are simultaneously evaluated:

- F_{LEN} preserves the basic shape of the hull;
- F_{MATCH} takes into account the distortion of the actual hull, in relation to the core affine transformation;

- F_{RATIO} evaluates the fraction of the total number of template quadtree nodes that have been successfully mapped onto the reference image. This is the final objective of the search which evaluates its overall success.

The results of the batch fitness evaluation are sorted in the increasing order of the fitness F_{LEN} values, and the top, i.e., the best fraction of the population of a specified size is selected and inserted in the pool B_P of the fitter solutions organized as a double-linked list. The operation of insertion is performed in such a manner that the pool remains sorted in the increasing order of fitness F_{LEN} values; and the fittest individuals are always placed on the top of the pool. If $\boldsymbol{V}^{\boldsymbol{curr}}$ is the current member of the pool, and $\boldsymbol{V}^{\boldsymbol{cand}}$ is a potential candidate for the insertion, then $\boldsymbol{V}^{\boldsymbol{cand}}$ is inserted in place of $\boldsymbol{V}^{\boldsymbol{curr}}$ only if the following conditions hold:

$$F_{LEN}^{cand} <= F_{LEN}^{curr}, \text{ and } F_{RATIO}^{cand} <= F_{RATIO}^{curr} . \quad (6.4)$$

Typically, some chromosomes in the best pool B_P are located close to each other; they can be clustered, to form one DSM simplex. Since each parameter domain is presented as a sequence of ranges, the algorithm can easily find the Hamming distance between any two chromosomes in the pool B_P. The pool is processed sequentially, starting from the top, i.e., from the fittest individual, and the matrix of the mutual Hamming distances is computed for all members of the pool. Two parameters control the clustering operation:

- maximum acceptable distance (threshold) L_R between two ranges, within which these ranges can be considered neighbors;

- maximum acceptable Hamming distance L_H (i.e., maximum number of ranges that are not neighbors) between two chromosomes in the best pool, for these chromosomes to be considered as vertices of the same DSM simplex.

Once the Hamming distance is computed, the neighboring chromosomes are clustered, to form DSM simplexes. Since the pool B_P is processed sequentially, starting from the top, the first vertex in the simplex is always guaranteed to be the fittest chromosome, which is considered basic, in relation to all its neighbors. If the Hamming distance between the basic chromosome and one of its neighbors is greater than zero, then those ranges of the neighbor that exceed the distance threshold L_R are replaced using the random mutation of the respective ranges of the basic chromosome, where the mutants are picked within the distance threshold L_R. If the number of chromosomes in the cluster is less than the number of vertices ($N + 1$), where N is the number of the affine parameters required by the DSM simplex, then additional chromosomes are created using random mutation of the basic chromosome.

Local search is conducted for all DSM simplexes generated from the best pool B_P. The quality of the individuals during the search is evaluated on the basis of their fitness F_{LEN}. At the end of the search, each simplex produces a chromosome with the lowest value of the fitness F_{LEN}. All such chromosomes are selected into the best pool B_{LOC}. If there is sufficient number of chromosomes, the matrix of Hamming distances is computed, and chromosomes are clustered, to form new DSM simplexes, in the same manner as it is done for generating simplexes from the pool B_P.

Since the best pool B_{LOC} is composed of the chromosomes that have the best fitness F_{LEN}, the local search is conducted for the new DSM simplexes using new fitness function F_{MATCH}. At the end of the global iteration, best pools B_P and B_{LOC} are sorted in the increasing order of the values of their respective fitness functions F_{LEN} and F_{MATCH}. The best chromosomes from the both pools are selected, to form the final best pool B_{FIN} which is sorted in the increasing order of the values of its fitness function F_{RATIO}. The specified number of chromosomes from the pool B_{FIN} is presented for the final evaluation.

The algorithm continues its work by evaluating the values of the hit function for all ranges of the parameter domains, accordingly assigning the ranges to the hit bins and generating new populations, in the manner described at the beginning of this section. The algorithm terminates when the final set of solutions retain its position within the specified number of iterations. Once the solutions are obtained for the hull transformations, the fittest chromosomes can be used as initial approximation for the refined local search phase which attempts to find the final piece-wise affine transformation of all sections of the image. The details of the local phase of the search are discussed in section 6.4.

6.3.1. Frontal Memory Management in Forming Population

The idea of mutation with memory discussed in Chapter 4 (section 4.4.2), is extended to the frontal memory management used to form a new population. The frontal algorithm of the memory management is presented in Figure 6.6.

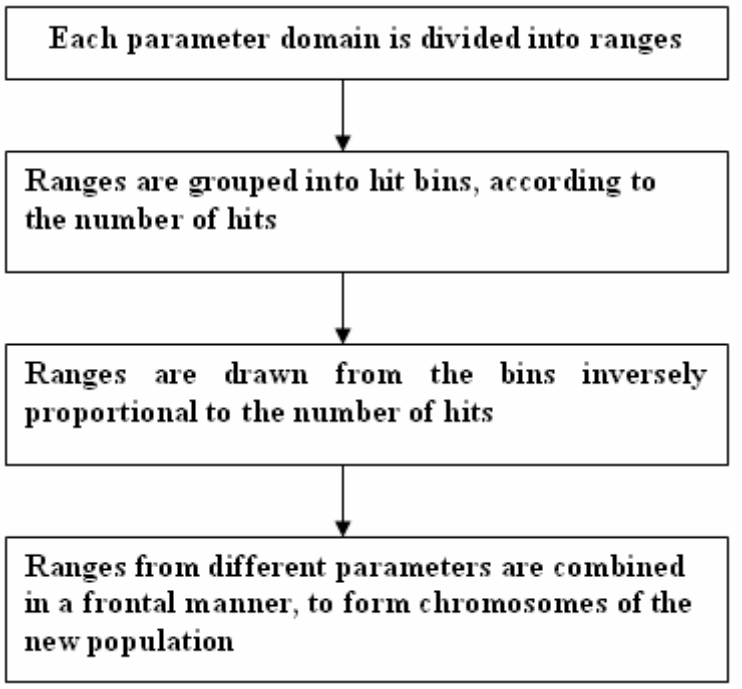

Figure 6.6: Frontal algorithm of memory management used to form a new population

Each parameter domain (i.e., *DX* domain, *DY* domain, etc.) is divided into ranges of the size $L = D / N$, where N is the number of ranges. The number of ranges is chosen in such a way that the local search procedure can find the optimum solution, if the latter is located within the limits of the range. Each range is defined by its left and right limits, its center, the hit function, and the bin, where the range is located. The center of the range serves as the representative value of the parameter component and is processed during the search. The hit function keeps track of the frequency of the range usage during all operations of crossover, mutation, and local search. Hit bins serve as containers that store the ranges. Each bin contains only ranges whose hit function takes value within the limits dynamically assigned by the algorithm to this bin.

At the beginning of every global evolutionary iteration, i.e., before forming the new population, all ranges are distributed among the hit bins, in accordance with the values of their hit functions. To form individuals of the new population, ranges are drawn from the bins inversely proportional to the values of their hit function, i.e., the largest proportion of the population is drawn from the bins with the least values of the hit function. This strategy insures that all ranges are nearly equally represented in the population. If some range *r* receives less attention at the current iteration *k*, i.e., has low value of its hit function, then at the next iteration (*k* + 1), the algorithm will place the range *r* into a bin that will have the largest draw rate, thus compensating for the under-usage of the range at the current iteration.

Under the mechanism of the fair representation of all ranges in the population, the same corresponding ranges from the different parameter domains will be drawn from the same corresponding bins and combined by the crossover operation, to form chromosomes of the new population. Therefore, every new population will be comprised of the same individuals. In order to break this dependency, a frontal approach is introduced in the following way. For each parameter, its respective set of hit bins is aligned, to form an infinite string. The bins from the different strings are combined, according to the following Formula:

$$B_{k,\,i} = B_{0,\,i} + k + l_i\,, \tag{6.5}$$

where $k = 0, 1,\ldots$ is the parameter order, i is the global iteration number, and l_i is the latency introduced by the iteration i. The latency can be set equal to the iteration number, in which case Formula (6.5) becomes

$$B_{k,\,i} = B_{0,\,i} + k + i\,. \tag{6.6}$$

Figure 6.7 shows the idea of the frontal algorithm, for a sample set of three parameters, each having four bins. As one can see, the bins from different parameter strings are aligned, thus forming a front that moves along the strings as the number of global iterations increases.

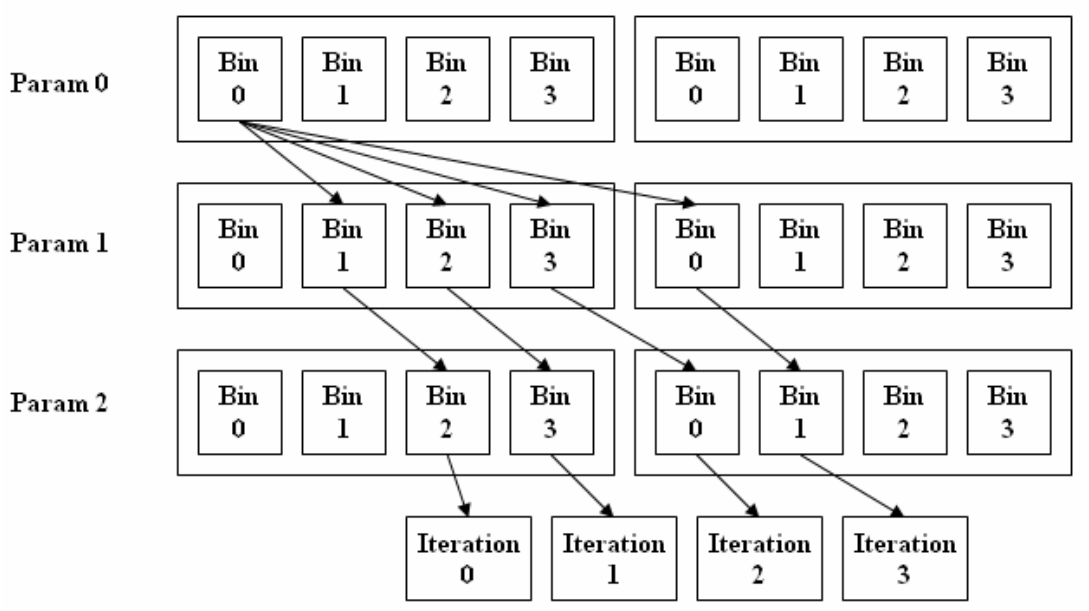

Figure 6.7: Frontal algorithm used to form chromosomes from the hit bins, in the case of three parameters and four hit bins

6.3.2. Multiple Objectives of the Global Hull Optimization

The global search algorithm attempts to find the closest match between the hull model H_T of the template image and the image of the response matrix M_R computed for the reference image; it does so by minimizing the difference between the two images. The quality of the chromosomes comprising the population is evaluated on the basis of three

objectives, i.e., fitness functions F_{RATIO}, F_{LEN}, and F_{MATCH}. These fitness functions are based on computing the following distances between the images H_T and M_R.

The optimum distance D_{RATIO} is defined as

$$D_{RATIO} = N_{optim} / N_{tot} , \tag{6.7}$$

where N_{tot} is the total number of nodes in the hull contour, and N_{optim} is the number of the optimum nodes that successfully matched the corresponding pixels of the response image M_R. Since the final goal of the optimization process is to find the complete match between the two images, the distance D_{RATIO} measures the degree of the ultimate success of the global hull optimization. The value $D_{RATIO} = 0$ indicates a complete failure of the search, and the value $D_{RATIO} = 1$ corresponds to the complete success, i.e., when all nodes of the hull have matched the response image M_R.

Elastic deformation D_{DEF} of the hull is defined as

$$D_{DEF} = \frac{\sum_{i=1}^{I} \left| \delta_{v,i} \right|}{N_{optim} F_A \delta_{\max}} , \tag{6.8}$$

where δ_v is the deviation of the matched node from the hull contour (i.e., the elastic deformation of the hull during the mapping process), I is the number of the deformed nodes, N_{optim} is the total number of the matched nodes, F_A is the area reduction factor, and δ_v is the specified maximum value of the deformation allowed by the search. The distance D_{DEF} evaluates the degree of the deviation of the matched hull contour from the hull contour of the image template.

The area reduction factor in Formula (6.8) is defined as the ratio of the areas of the matched hull contour A_{MATCH}, and the template contour A_T, i.e.

$$F_A = A_{MATCH} / A_T \,, \tag{6.9}$$

and plays an important role in fitness evaluation, to ensure that we obtain the largest possible match during the search.

The length distance D_{LEN} is defined as

$$D_{LEN} = \frac{\left| F_A \sum_{j=1}^{N_{optim}} l_j - 1.5 L_{HC} \right|}{L_{HC}}, \tag{6.10}$$

where F_A is the area reduction factor, l_j is the segment length between the matched nodes j and $(j + 1)$ of the hull contour, N_{optim} is the total number of the matched contour nodes, and L_{HC} is the total length of the hull contour of the image template. The length distance D_{LEN} evaluates the degree of preservation of the hull contour of the image template during the search.

The area distance D_{AREA} is defined as

$$D_{AREA} = \frac{| A_M F_A - A_H |}{A_H} \,, \tag{6.11}$$

where A_M is the number of the black nodes enclosed inside the matched hull, F_A is the area reduction factor, and A_H is the area of the template hull defined as the number of the black nodes inside the template hull. The area distance D_{AREA} evaluates the degree of preservation of the inner area of the template hull during the search.

The fitness function F_{RATIO} is defined as

$$F_{RATIO} = 1 - D_{RATIO} , \tag{6.12}$$

and serves as the main objective function and the ultimate goal of the global hull optimization. If all contour nodes of the hull have matched for a chromosome $\boldsymbol{V}$, then its fitness takes the minimum value $F_{RATIO} = 0$. If none of the nodes have matched, then the fitness value equals to $F_{RATIO} = 1$. Although F_{RATIO} is the main objective of the global hull optimization, it cannot be directly used during the search, because the latter will attempt to find the largest possible contractive transformation, i.e., it will attempt to map the template hull onto the smallest spot in the reference image. This is the reason why the fitness F_{RATIO} is used only in the final stage of the algorithm, as shown in Figure 6.5, to evaluate and rank the final set of the best solution.

The fitness function F_{LEN} is one of two working optimization objectives that are intensively used during the intermediate process of the search; it is defined as

$$F_{LEN} = D_{LEN} + D_{AREA} . \tag{6.13}$$

The main goal of F_{LEN} is to preserve, as much as possible, two main features of the hull shape – its contour length L_{HC}, and its area A_H. The search for the minimum value of the function F_{LEN} always attempts to find the largest, or the outmost matching hull, thus preventing the process from converging into a small spot in the image M_R.

Once the pool B_{LOC} of the fittest chromosomes has been filled with the chromosomes that provide the largest matching hulls, the search switches to the new objective, and attempts

to find the closest and least deformed solution within the pool B_{LOC}. The objective of the optimization now is to minimize the fitness function F_{MATCH}

$$F_{MATCH} = \frac{\left| N_{optim} F_A - N_{tot} \right|}{N_{tot}} + D_{DEF} \ . \tag{6.14}$$

The function F_{MATCH} has two components: the first component evaluates the degree of the match, while the second component evaluates the elastic deformation of the matched hull, in relation to the original template hull. Therefore, the search attempts to maximize the degree of the match and simultaneously to place the matched hull in the closest possible way to the original hull, reducing the deviation from the latter.

The approach to multi-objective optimization undertaken in the algorithm of the global hull optimization shown in Figure 6.5 is to utilize three different objectives, in the form of three fitness functions, in the consecutive manner. First, the pool of the largest matching hulls is found using the fitness function F_{LEN} that ensures the preservation of the main shape features of the template hull. Then, the obtained fittest individuals are refined with the fitness function F_{MATCH} which identifies the best matched solutions that are least deformed, in relation to the original template hull. Finally, the best individuals obtained with the objectives F_{LEN} and F_{MATCH}, are evaluated and ranked according to the main final objective of the search, the fitness function F_{RATIO}. The latter allows the algorithm to select the best solutions according to the number of matching nodes. The final set of solutions is presented to the decision maker and, if needed, participates as initial approximation in the second pass of the algorithm – the local search for the

optimum piece-wise transformation of the entire template image. This second pass is presented in the next section.

6.4. Local Optimization of the Piece-Wise Affine Transformation

The optimum final solutions found by the global hull optimization serve as the initial approximation for the local optimization of the entire template image. Local optimization attempts to find a piece-wise affine transformation that can produce the best mapping from the template image onto the reference image in the response space. The template image is represented as a set of sections of specified dimensions, i.e., $Img_T = \{S_1, S_2, \ldots, S_M\}$, where M is the number of sections. Then the sought transformation T of the entire template is represented in the form of superposition of the affine transformations A_k of the individual sections S_k:

$$T = \sum_{k=1}^{M} A_k \ , \tag{6.15}$$

where each individual affine transformation A_k is defined by the parameter vector

$$\boldsymbol{V_k} = (DX_k, DY_k, \theta_k, SX_k, SY_k, SHX_k, SHY_k) \ . \tag{6.16}$$

The overall strategy of the multi-objective local optimization of the piece-wise affine transformation of the template image in the response space is presented in Figure 6.8. The template response image is broken down into sections that form a hierarchical tree structure. The sections that are the closest to the template hull are placed at the top of the hierarchy; they form the base level of the tree. The sections that are adjacent to the base level form the next level of the tree. The process of building the sections tree continues until all template sections are placed on the proper levels of the tree – see section 6.4.1.

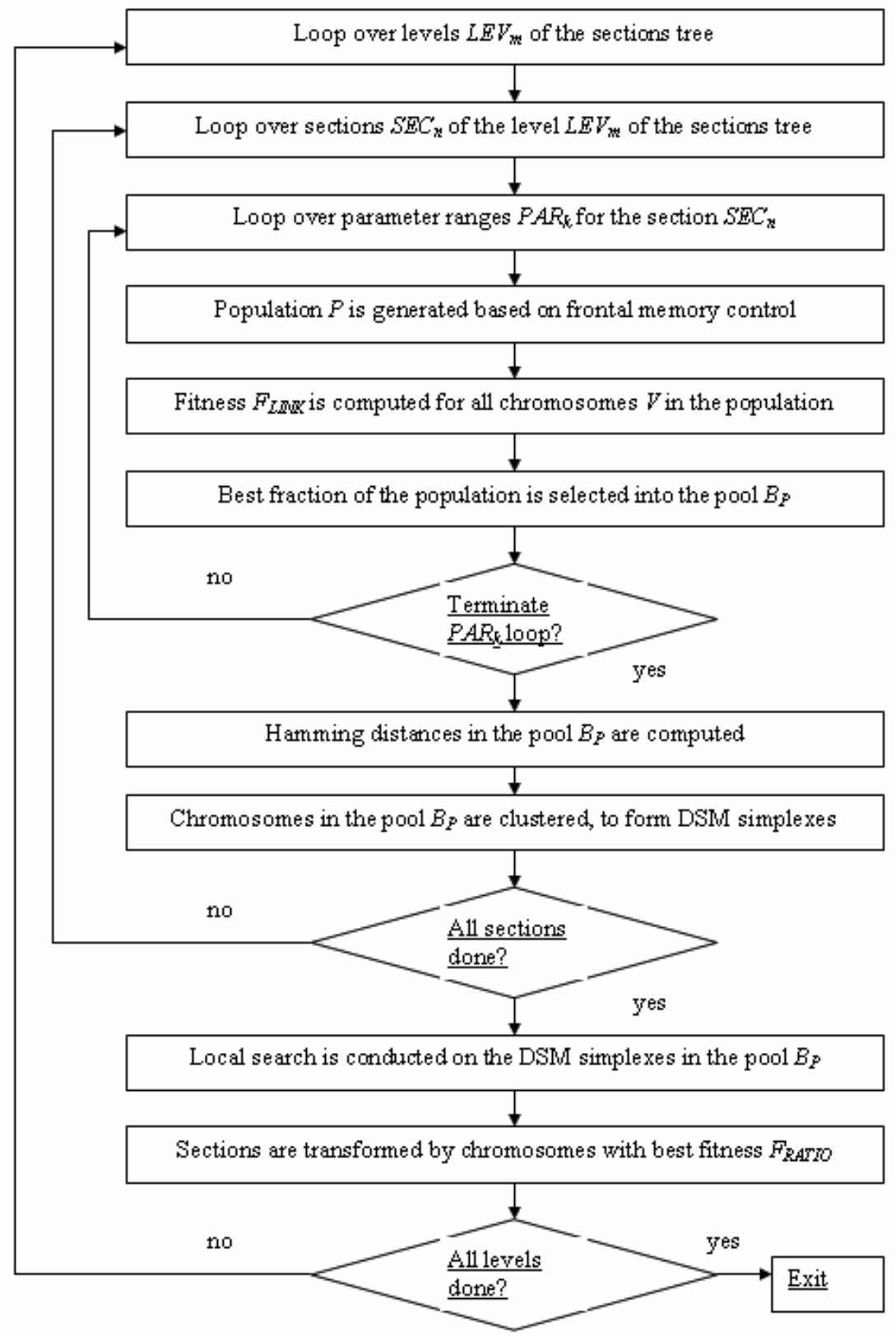

Figure 6.8: Algorithm of the local optimization of the piece-wise affine transformation

Once the tree is completed, the algorithm processes all sections sequentially, level by level, in a “peeling-off” manner, starting from the base level at the top of the hierarchy and moving down the tree. The rational behind this “peeling-off” technique is that the base sections adjacent to the hull will have their transformation vectors close to the hull transformation; these vectors can be relatively easily found using the hull transformation as the initial approximation. Once the transformation of the base level has been found, the base sections are correspondingly transformed and eliminated from the further work of the algorithm, i.e., they are “peeled off”. All the following levels of the tree have to be adjusted, level by level, following the transformations of the base level. Next, the sections adjacent to the base level can be considered. Since they are close to the base level, it is logical to assume that their transformation vectors will be close to the transformation vectors of the corresponding base sections; therefore, the latter can serve as the initial approximation for the optimization of the adjacent sections. The entire “peeling-off” process, therefore, works as described below.

Step 1: Parental sections are optimized using the transformation vectors of the grandparents as the initial approximation, and are eliminated (“peeled off”) from the further search.

Step 2: Offspring sections are adjusted according to their parents’ transformations; the adjustment process propagates throughout the entire tree.

Step 3: Offspring are optimized using the transformation adjustment as the initial approximation for their own transformations, and are eliminated ("peeled off") from the further search.

Step 4: The "peeling-off" process continues until all sections have been processed.

The final transformation $A_{S,LEV}$ of any individual section S_k located at some level *LEV* of the sections tree, therefore, is the combination of all transformations of this section occurred at the previous levels:

$$A_{S,LEV} = A_{S,LEV-1} \times A_{S,LEV-2} \times \cdots \times A_{S,0} \ . \tag{6.17}$$

Since at any level of the section tree, there is a fairly good initial approximation of the piece-wise affine transformation of the sections inherited from the parental level, it is logical to assume that the offspring transformation vectors do not deviate too far from their parents. Therefore, the initial population for the optimization at the current tree level *LEV* is initially generated within a small range around the parental parameters. If the optimization search does not produce good results, the new population is generated using a larger parameter range. The process of expanding the parameter range continues until no more good solutions have been produced. The process of expanding parameter ranges around their initial approximation is, in fact, the process of relaxation of the initial constraints imposed on the parameter range. The population for every parameter range is generated using the same frontal memory management procedure that is discussed in section 6.3.1 of this Chapter.

Two fitness functions, F_{LINK} and F_{RATIO} are used as two objectives of the local piece-wise optimization. Once the new population has been generated, the quality of every chromosome is evaluated on the basis of the fitness functions F_{LINK}. This function takes into account the relative position of the section among its adjacent parents and siblings, thus preserving the integrity of the entire template model. The detailed discussion of the fitness functions F_{LINK} and F_{RATIO} is presented in section 6.4.2.

The best fraction of the current population is selected and inserted into the best pool B_P. Make a note that the pool collects the best solutions obtained for the current section, over all expansion of its parameter range. The matrix of Hamming distances is computed for all chromosomes in the pool, and DSM simplexes are generated by clustering the neighboring individuals in the manner discussed earlier in section 6.3.

Once all sections of the current tree level *LEV* have been processed, local optimization is performed for all DSM simplexes using the fitness function F_{LINK}. The best final solutions are selected and ranked according to the final objective of the search defined as the fitness function F_{RATIO}. The latter evaluates the degree of the match between the sections of the template and the corresponding sections of the reference image, in the response space. All sections of the current tree level *LEV* are transformed, each according to its respective best final solution $\boldsymbol{V_S}$.

Since the sections of the level *LEV* have undergone affine transformations A_S, all the following levels of the sections tree have to be accordingly adjusted, i.e., transformed

using the transformations A_S, in the following way. If a section S at the level LEV has K parents (i.e., adjacent sections) at the previous level ($LEV - 1$), then its transformation vector $\boldsymbol{V_{S,LEV}}$ is computed as the weighted mean of all parental transformation vectors $\boldsymbol{V_{k,LEV-1}}$, i.e.,

$$V_{S,LEV} = \frac{\sum_{k=1}^{K} w_k V_{k,LEV-1}}{\sum_{k=1}^{K} w_k} , \tag{6.18}$$

where w_k is the weight assigned to the parental transformation vector $\boldsymbol{V_{k,LEV-1}}$; it is inversely proportional to the vector's fitness

$$w_k = 1 / F_{RATIO,,k} , \tag{6.19}$$

Formula (6.19) simply states that the parental chromosomes of higher quality will have a greater contribution in the offspring transformation. The procedure of the local optimization of the piece-wise transformation of the template image terminates when all sections, at all levels of the sections tree have been processed. The best final piece-wise transformations of the sections of the template image are presented to the decision maker for the final evaluation.

6.4.1. Tree Structure of the Template Image

The template image is broken down into sections which are organized in a hierarchical manner, to form a sections tree. The top level of the tree is comprised of the chromosomes that are adjacent to the template hull. In a sample template image shown in Figure 6.9, sections 0 through 8 are the top-level base sections which are processed first by the local optimization procedure. To find their transformation vectors $\boldsymbol{V_S}$, the

transformation *T* of the template hull shown in Figure 6.4, is used as the initial approximation, i.e., the starting point of the search. Sections 9 through 14 form the second level of the tree, etc. The entire sections tree for the sample template is presented in Figure 6.10.

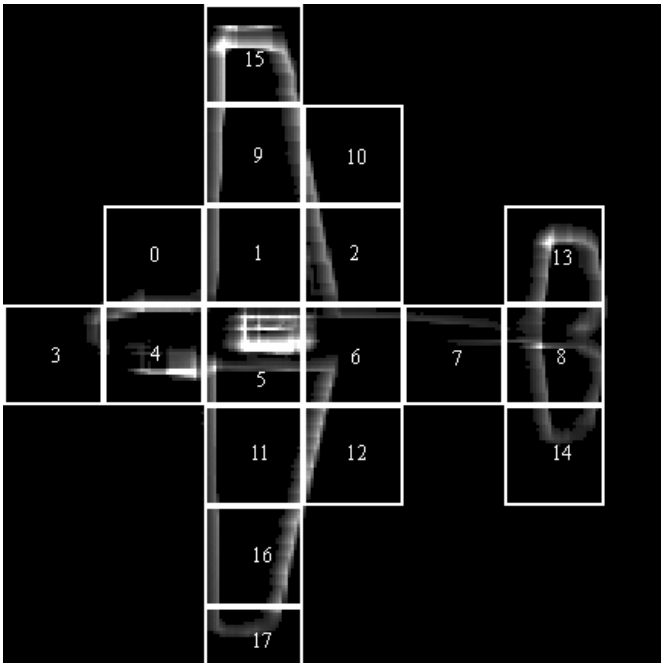

Figure 6.9: Sample template response image with the indicated image sections

When sections 0 through 8 of the base level have been optimized and transformed according to their best final transformation vectors, sections 9 through 14 of the next level are transformed using the weighted mean of their parents' transformation vectors. For example, the transformation vector $\boldsymbol{V_9}$ of section 9 is obtained as follows:

$$V_9 = \frac{w_0 V_0 + w_1 V_1 + w_2 V_2}{w_0 + w_1 + w_2} \ , \tag{6.20}$$

where the weighted factor is computed as the inverse of the corresponding value of the sections' fitness function, i.e., as

$$w_0 = 1 / F_{RATIO,0} \, , \, w_1 = 1 / F_{RATIO,1} \, , \, w_2 = 1 / F_{RATIO,2} \, . \tag{6.21}$$

When optimization starts for sections 9 through 14, their adjusted transformations become the initial approximation of their new transformation vectors.

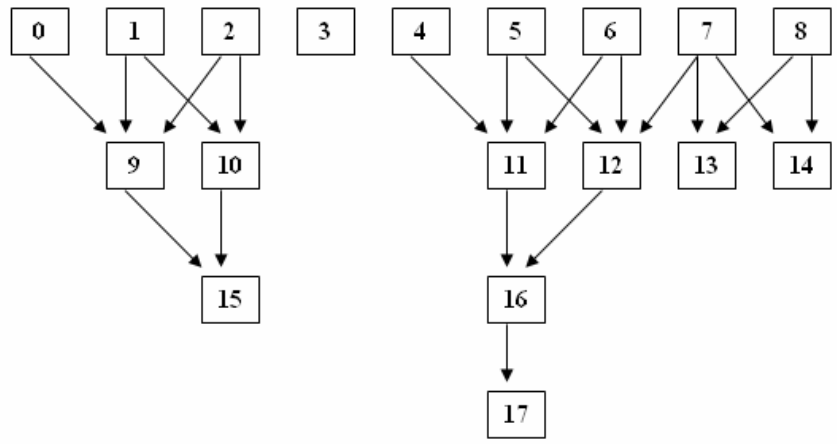

Figure 6.10: Sections tree of the sample template image

6.4.2. Multiple Objectives of the Local Piece-Wise Optimization

In a manner similar to the global search, the final goal of the local optimization is to maximize the ratio of the matched nodes of the template which is represented by the collection of sections organized in a tree structure. Therefore, the main objective of the search for each section is stated in the form of the fitness function F_{RATIO} defined similarly to the global search, as follows:

$$F_{RATIO} = 1 - D_{RATIO} \, , \tag{6.22}$$

$$D_{RATIO} = N_{optim} / N_{tot} \, , \tag{6.23}$$

where D_{RATIO} is the optimum distance, N_{optim} is the number of the optimum (i.e., matched) nodes, and N_{tot} is the total number of nodes in the section.

In the search practice, however, the objective F_{RATIO} can rarely be used. Even under the assumption that the transformation of every previous level of the sections tree can serve as the initial approximation for the next tree level, the sections of this level can still be significantly deformed, which can result in the erroneous match. To reduce the risk of such mismatch, the working objective of the local optimization is constructed in such a way that it preserves the integrity of the entire template by asserting the coherent placement of the transformed section, in relation to its neighbors. The fitness function F_{LINK} corresponding to the objective of the optimization is defined as

$$F_{LINK} = \frac{\left| N_{optim} F_A - N_{tot} \right|}{N_{tot}} + E_L + E_R \ , \tag{6.24}$$

where N_{optim} is the number of the optimum nodes of the section, N_{tot} is the total number of nodes in the section, F_A is the area reduction factor of the section, similar to the area reduction factor for the hull defined in section 6.3.2, E_L is the energy of the linear deformation, and E_R is the energy of the angular deformation of the section. Here, the first term, essentially, evaluates the ratio of the successfully matched nodes of the section.

The second and the third terms in Formula (6.24) ensure that the current section is coherently placed among the surrounding neighbors, i.e., its parental sections and siblings. The energy E_L of the linear deformation is defined as the weighted mean displacement of the section vertices occurred after the transformation, from the respective vertices of the neighboring sections:

$$E_L = \frac{\sum_{u=1}^{U} \sqrt{\Delta_{x,u}^2 + \Delta_{y,u}^2}}{UL_S D_{RATIO}} \ , \tag{6.25}$$

where U is the number of section vertices connected to the section neighbors, L_S is the mean value of the section linear dimension, D_{RATIO} is the optimum distance, $\Delta_{x,u}$ is the weighted mean of the displacement of the vertex u in the x direction defined as

$$\Delta_{x,u} = \frac{\sum_{k=1}^{K} \delta_{x,k} w_k}{\sum_{k=1}^{K} w_k} \ , \tag{6.26}$$

$\Delta_{y,u}$ is the weighted mean of the displacement of the vertex u in the y direction defined as

$$\Delta_{y,u} = \frac{\sum_{k=1}^{K} \delta_{y,k} w_k}{\sum_{k=1}^{K} w_k} \ , \tag{6.27}$$

and $\delta_{x,k}$ and $\delta_{y,k}$ are the respective x and y displacements of the vertex u, in relation to the corresponding vertices of the section neighbor k. The weights w_k are computed in accordance with Formula (6.19), such that the neighbors of a higher quality make a greater contribution to the energy E_L of the section and act as its strong attractors.

The energy E_R of the angular deformation of the section is defined in a similar way, i.e.

$$E_R = \frac{\sum_{u=1}^{U} \sqrt{\Psi_{c,u}^2 + \Psi_{s,u}^2}}{UD_{RATIO}} \ , \tag{6.28}$$

where $\Psi_{c,u}$ is the weighted cosine of the angular deformation of the vertex u defined as

$$\Psi_{c,u} = \frac{\sum_{k=1}^{K} \psi_{c,k} w_k}{\sum_{k=1}^{K} w_k}, \tag{6.29}$$

$\Psi_{cs,u}$ is the weighted sine of the angular deformation of the vertex u defined as

$$\Psi_{s,u} = \frac{\sum_{k=1}^{K} \psi_{s,k} w_k}{\sum_{k=1}^{K} w_k}, \tag{6.30}$$

and $\psi_{c,k}$ and $\psi_{s,k}$ are the respective cosine and sine of the angular deformation of the vertex u, in relation to the corresponding vertices of the section neighbor k.

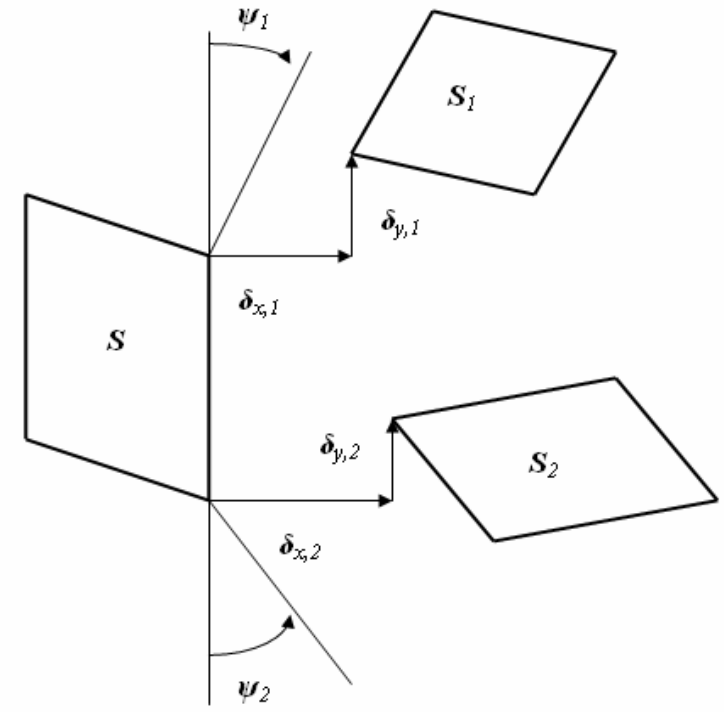

Figure 6.11: Linear δ and angular ψ deformations of the section S, in relation to its neighbors S_1 and S_2

Figure 6.11 illustrates linear and angular deformations of a sample section *S* that has two transformed neighbors, S_1 and S_2. Because sections are optimized in the consecutive manner, the first section of the current tree level *LEV* does not have siblings that have already been optimized on this level, and its deformation energy can only be evaluated in relation to its parents. In order to compensate for the insufficient information, all sections on the level *LEV* are sorted and ranked in the increasing order of their fitness values. The fittest section that has the lowest fitness value is selected as the starting section of the local search at the level *LEV*.

6.5. Computational Experiments

This section illustrates the application of the multi-population, multi-objective optimization algorithm described in the previous sections of this Chapter, to solving some difficult problems of imaging optimization. Each problem involves a piece-wise mapping of two template images of the same object taken from different camera views, onto the same reference image. The first problem discussed in section 6.5.1, considers human body registration in the still photo of a street scene. Recognition of a 3-dimensional object shape in the 2-dimensional space is presented in section 6.5.2. A sample elastic registration of medical images is shown in section 6.5.3.

6.5.1. Human Body Registration

All the algorithms of human body registration with EAs mentioned in Chapter 3 (section 3.4.6) use various techniques based on motion analysis, in order to extract human body from video sequences. Motion analysis effectively reduces the computationally hard

problem of global optimization to a much more feasible problem of local optimization, by spotting the location of the body, thus drastically reducing the size of the search space. In the case, when motion analysis is infeasible or unavailable, e.g., in the recognition and registration of still images, the entire space of the potential parameter vectors $\boldsymbol{V}$ has to be utilized for the search. But this is exactly the kind of a problem where EAs are particularly efficient. Human body registration with the global pass of the multi-population, multi-objective hybrid EA model and image response analysis presented earlier in this Chapter, works in the following way.

Step 1: Body templates Img_{Bn} (n = 1,…, N, where N is the number of the object views) are aligned (approximately), with respect to each other, using the same value of the angle θ; and shear factors *SHX* and *SHY*.

Step 2: Response matrices R_R of the reference image Img_R, and R_{Bn} of the template images Img_{Bn}, are computed. Make a note that this can be done at the pre-processing stage, before the evolutionary search begins.

Step 3: Response matrix R_{Bn} of each template is decomposed into quadtrees. Decomposition significantly reduces the total amount of information that has to be processed during the search, because only quadtree nodes participate in the fitness evaluation for any trial chromosome $\boldsymbol{V}$. This transformation can also be done at the pre-processing stage.

Step 4: A hull H_{Bn} (preferably, convex) is defined for each of the templates Img_{Bn}. The hull is interactively defined by the user, and has to preserve the main shape of the body. The template hull can be defined at the pre-processing stage and stored in a database, as a part of the quadtree decomposition performed in step 3.

Step 5: Evolutionary search with HEA is performed in the response space.

The entire chromosome population TP consists of N subpopulations SP_n, each of which corresponds to the particular template Img_{Bn}; all subpopulations are aligned with each other. Subpopulations are fully and independently evaluated on the basis of their chromosomes' fitness, where fitness F_{LEN} of the individual chromosome $\boldsymbol{V}$ is defined as the degree of the match between the main features of the corresponding template hull H_{Bn} and the response matrix R_R of the reference image. The chromosomes that simultaneously provide lower (i.e., better) fitness values for all subpopulations, are selected into the elite pool B_P. Once in the pool B_P, elitist chromosomes are clustered, in accordance with their mutual Hamming distance. Concurrent local search using a version of Downhill Simplex Method is conducted in such a way that each of the clusters forms an individual simplex. Thus, the number of the concurrently evaluated simplexes equals the number of the clusters in the subpopulation. The refined elitist chromosomes are selected into the second elitist pool, B_{LOC}, and the DSM search is performed using the second search objective, F_{MATCH}. When the search terminates, the best elitist chromosomes computed throughout all iterations are ranked, in accordance with their corresponding fitness values

F_{RATIO}, i.e., with the degree of the match between the template and the reference image. The list is presented for the final evaluation of the decision maker.

In order to validate the proposed algorithm of human body registration with HEA and image response analysis, computational experiments are conducted with a set of 2-D grayscale images shown in Figure 6.12. The test set includes a 512×384-pixel image of a street scene (reference image Img_R), which includes a figure of a moving person; and two 128×256-pixel images of different people (template images Img_B) taken from different camera views, i.e., from the back, and from the side. The templates are approximately aligned with each other, with respect to their vertical angular position; however, the templates have different scaling factors. The problem of human body registration is stated as the global optimization problem of finding the optimum vector $\boldsymbol{V}$ minimizing the difference F between the template images Img_B and reference image Img_R. The size of the population equals 64, and the ranges of the sought parameters are shown in Table 6.1.

Figure 6.12: Reference image of a street scene (left); human body templates (right)

Parameter	***θ***	***DX***	***DY***	***SX***	***SY***	***SHX***	***SHY***
Lower limit	- 30°	0	0	0.3	0.3	- 0.5	- 0.5
Upper limit	+ 30°	512	384	1.0	1.0	+ 0.5	+ 0.5

Table 6.1: Parameter ranges used in the human body registration

Following the algorithm of the multi-population, multi-objective optimization described in sections 6.2 through 6.4 of this Chapter, the response matrices for all images are computed. The images of the response matrices are shown in Figure 6.13. For each template, a hull H_B is entered interactively – see Figure 6.14. A screen snapshot presented in Figure 6.15 shows the results of the quadtree decomposition and the hulls of both templates, before the evolutionary search begins (in the pre-processing stage). A screenshot in Figure 6.16 shows the intermediate registration results after the first iteration of the algorithm.

Figure 6.13: Response matrices of the reference image (left) and templates (right)

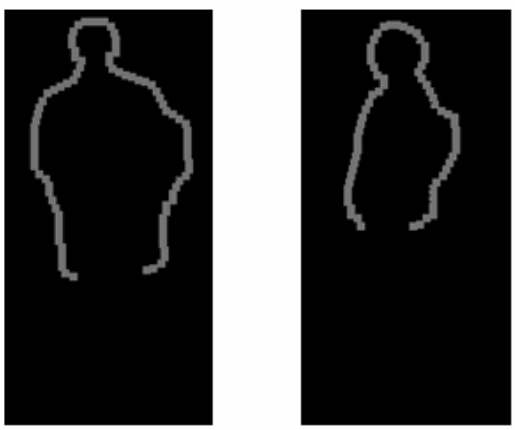

Figure 6.14: Sample hulls corresponding to the template images

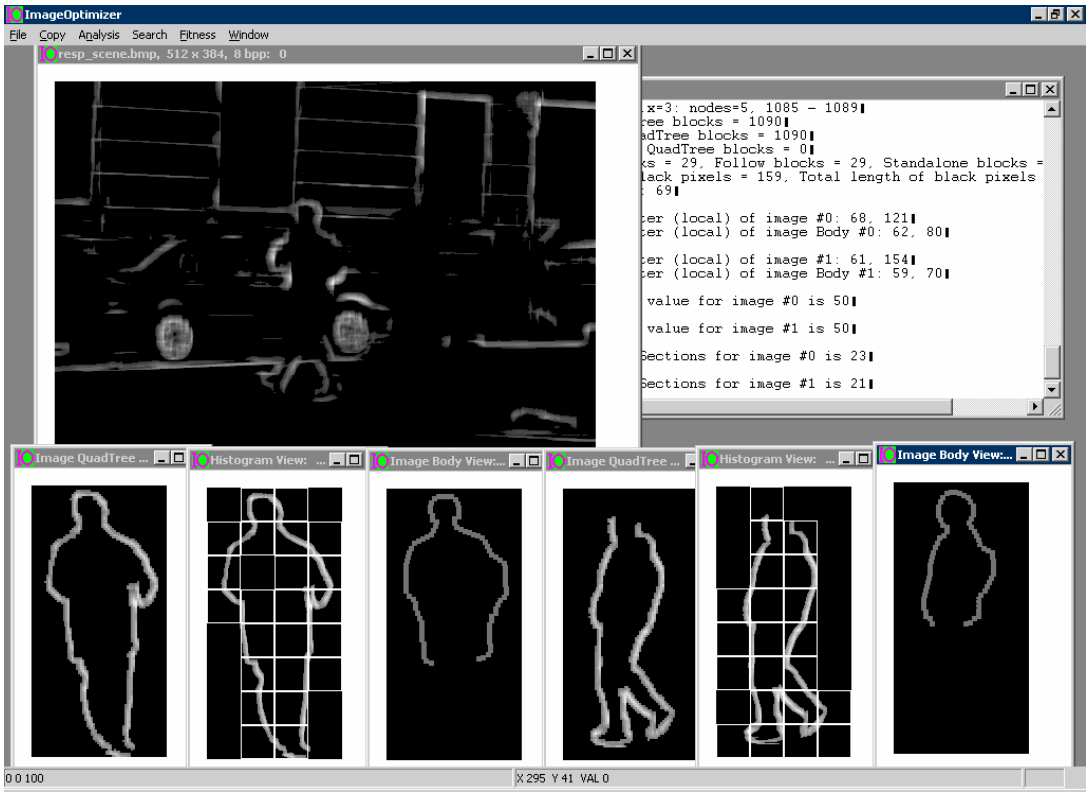

Figure 6.15: Screenshot after the pre-processing stage showing the quadtree decomposition and the template hulls

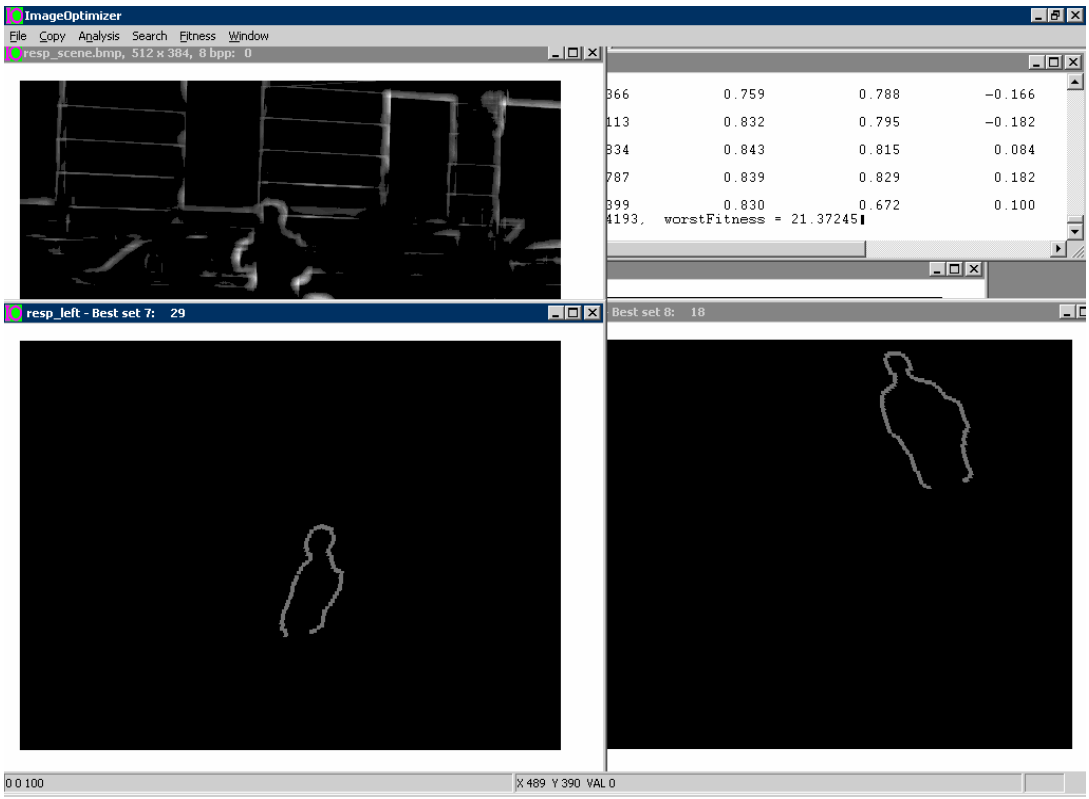

Figure 6.16: Screenshot after the first iteration of the evolutionary search

Figure 6.17: Final results of the human body registration: the reference image (left) and the registered templates (center and right)

The final results of the registration after iteration 6 are shown in Figure 6.17. Table 6.2 provides the numerical values of the registration parameters, together with the degree *r* of the match between the images. Make a note that larger values of *r* correspond to a better

match. As one can see, the proposed algorithm is able to find satisfactory results within a relatively short span of time. The best match $r = 0.797$ (approximately 80%) occurs between the reference image and the template corresponding to the side image of a person, while the template corresponding to the back image displays a slightly worse match of only $r = 0.670$ (i.e., 67%). One can conclude that a picture of a person in the scene is most likely taken from the side view, which can also be seen from the visual evaluation of the image.

Parameter	θ	***DX***	***DY***	***SX***	***SY***	***SHX***	***SHY***	***r***
Template 1 (back)	0.4°	159	108	0.91	0.91	- 0.13	- 0.02	0.670
Template 2 (side)	- 16.5°	179	83	0.78	0.77	- 0.09	- 0.03	0.797

Table 6.2: Registration parameters for the human body registration

6.5.2 Recognition of a 3-D Object in the 2-D Space

Direct application of the evolutionary search strategy to the mapping of 3-D objects encounters at least two serious problems:

1. The object in the scene can be arbitrarily rotated in the 3-D coordinate space, while available template images typically represent only one, or a few 2-D viewpoints of the object. Consequently, a single transformation $A(\boldsymbol{V})$ can no longer correctly represent the sought mapping between the images, under the condition of incomplete information.
2. Direct comparison of image pixel values frequently cannot be used, since the different viewpoints might result in the different pixel distributions. Moreover,

during the search for a viable transformation $A(\boldsymbol{V})$ between two images, Img_0 and Img_1, the pixels of the image Img_1 might be erroneously mapped into the different pixels of the image Img_0.

The first of the above mentioned problems suggests using a set of image transformations $SA = \{A(\boldsymbol{V}k)\}$, rather than a single transformation, in such a way that each individual transformation $A(\boldsymbol{V}k)$ in the set SA is applied to a corresponding individual section of the image Img_1. In other words, $A(\boldsymbol{V})$ becomes a piece-wise approximation of the actual 3-D mapping of the object. The original optimization problem of finding the parameter vector $\boldsymbol{V}$ minimizing the difference between the images turns into a multi-objective optimization problem (MOOP) of finding a set of feasible parameter vectors $\boldsymbol{V}_k$ minimizing the differences between the individual sections of the images Img_1 and Img_0. The second problem requires abandoning the idea of the pixel-wise comparison of the actual images and relocating the search into a suitable feature space. As shown in Chapter 5, using Image local response proves to be a viable alternative in the HEA-based search for a proper mapping of 2-D grayscale images.

The possibility of extending the HEA-based approach to the 2-D object mapping is tested on a set of three grayscale images shown in Figure 6.18. The 300×300-pixel reference image Img_0 contains an object arbitrarily rotated in the 3-D coordinate system. The template images are the 178×195-pixel top view Img_1, and the 185×66-pixel left view Img_2 of the same object.

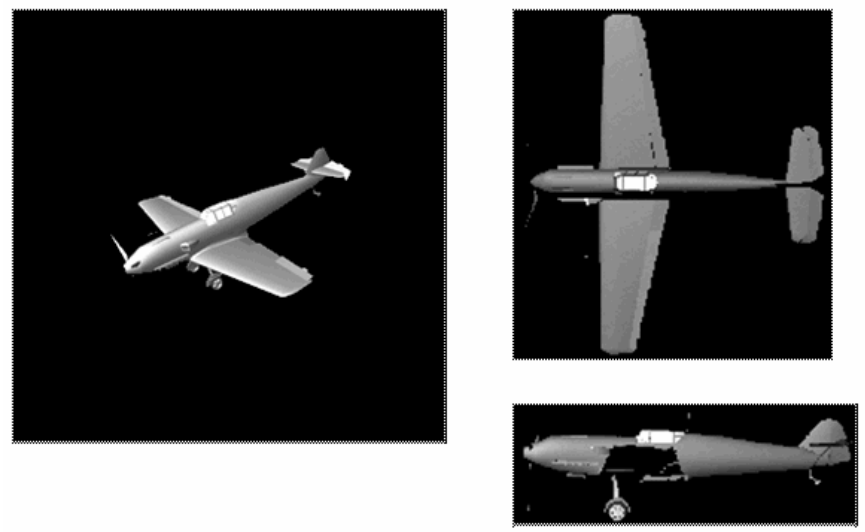

Figure 6.18: Image of the scene with a 3-D object (left); the template images (right)

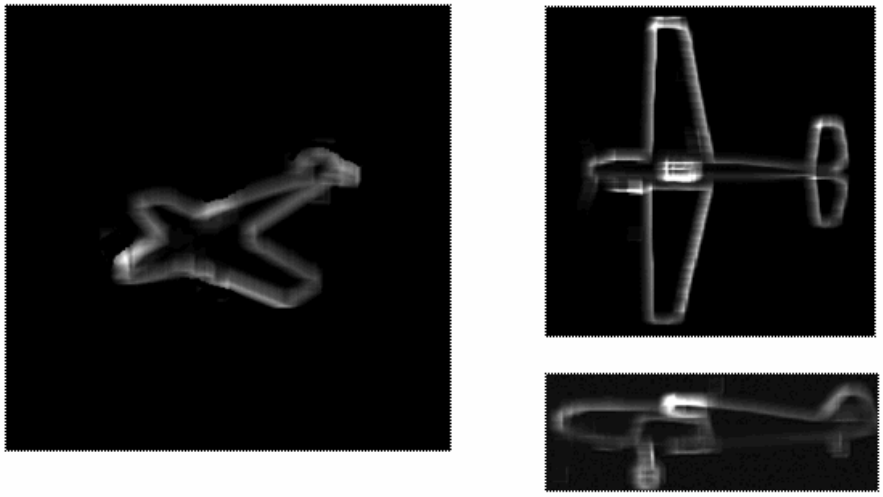

Figure 6.19: Response matrices of the scene (left) and template images (right)

Figure 6.19 shows response matrices M_R for all three test images, under the general affine transformation defined by the 7-dimensional vector of parameters $\boldsymbol{V} = \{DX, DY, \theta, SX, SY, SHX, SHY\}$. As one can see, the response values highlighted in Figure 6.19 emphasize the sections of the image that undergo the most noticeable changes after the

transformation $A(\boldsymbol{V})$ has been applied to the image. The number of the response values participating in the operations of image comparison is effectively reduced by computing the histogram of the template image response, and by its further piece-wise approximation with a quadtree. Finally, the quadtree nodes are approximated by their mean gray values; the latter are compared against the response values of the 2-D object in the scene.

The search for the proper mapping from the template images onto the scene starts with the global pass of the algorithm, in order to find the elastic affine transformation of the template hulls shown in Figure 6.20, alongside with the template sections. The parameter ranges of the global search are presented in Table 6.3.

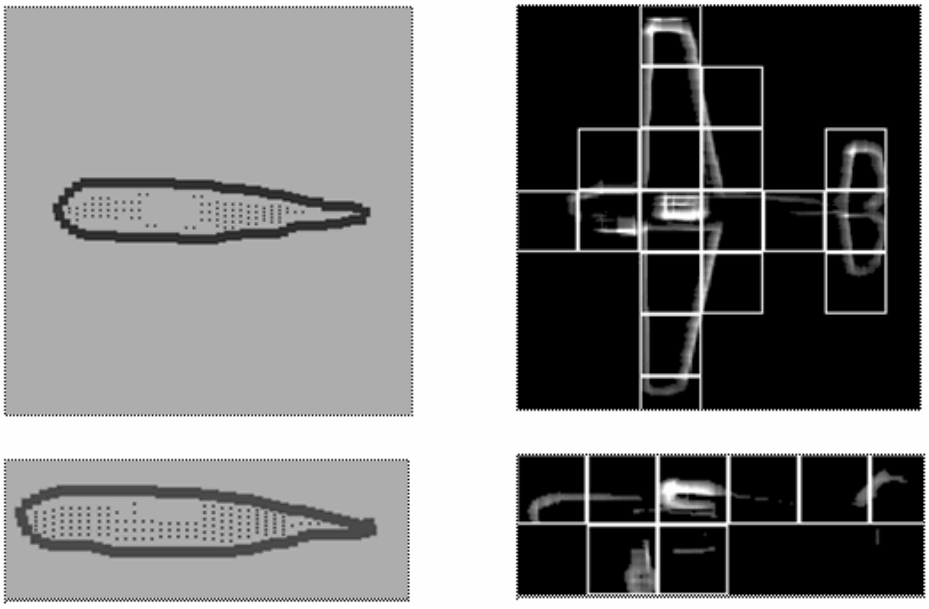

Figure 6.20: Template hulls and image sections: the top (top) and the side (bottom) views

Parameter	***θ***	***DX***	***DY***	***SX***	***SY***	***SHX***	***SHY***
Lower limit	- 90°	0	0	0.3	0.3	- 0.5	- 0.5
Upper limit	+ 90°	300	300	1.0	1.0	+ 0.5	+ 0.5

Table 6.3: Parameter ranges used in the global search for the 3-D object recognition

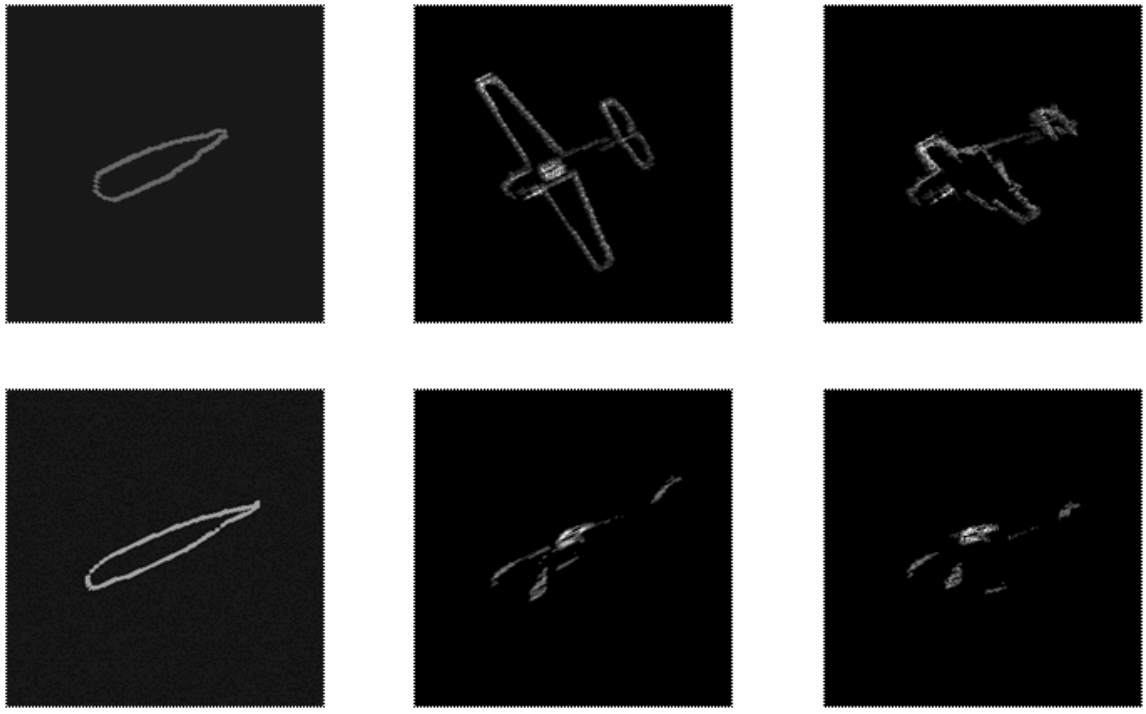

Figure 6.21: Results of three different phases of the algorithm for the top (top row) and the side (bottom row) views: the global transformations of the hulls (left column); the initial transformation of the template sections (center column); and the final piece-wise transformation of the template sections (right column)

Figure 6.21 shows the results of three different phases of the algorithm: the global affine transformation of the template hulls, the respective transformation of the template sections, and the final piece-wise transformation of the template sections. As one can see,

the global pass of the algorithm is able to find a fairly good approximation to the actual elastic transformations of the template hulls, including their mutual angular alignment. The second, local pass finds a good set of transformation vectors for the template sections – compare the rightmost column in Figure 6.21 with the original image in Figure 6.19 (left).

6.5.3 Elastic Registration of Medical Images

Registration of series of medical images obtained from MIR or CT-scan is another example of a difficult problem in imaging optimization. Different slices of the same organ from the scan have to be mutually registered. The difficulty stems from the fact that the organ changes its shape between the slices. Therefore, the registration algorithm has to be able to deal with these changes; it has to find the correct mappings between the slices. The computational experiments in this section show the work of the two passes of the algorithm, on elastic registration of a sample set of three images shown in Figure 6.22. One of the images (Figure 6.22, left) plays a role of the reference image, while the other two images (Figure 6.22, center and right) serve as templates that have to be mapped onto the first image.

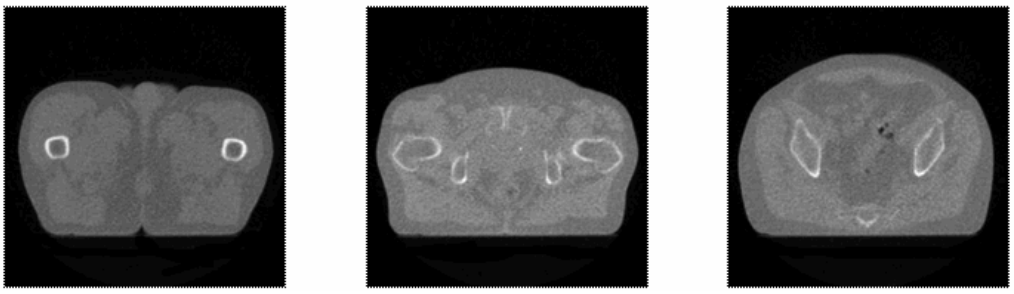

Figure 6.22: Original medical images subject to elastic registration

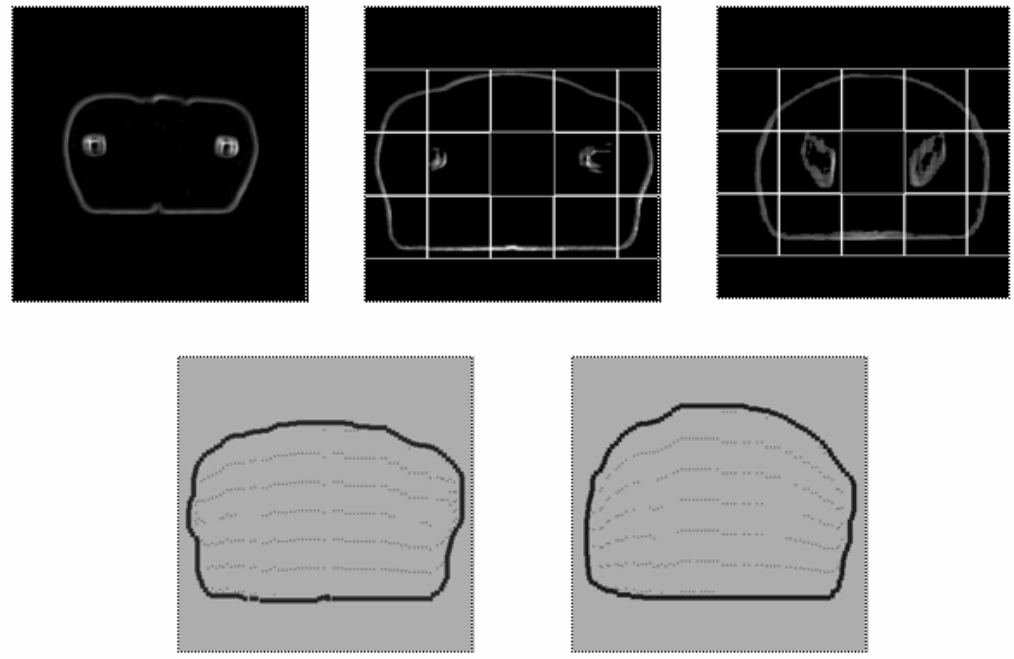

Figure 6.23: Local responses of the reference (top row, left) and templates (top row, center and right); images of the template hulls (bottom row)

Image local response is computed for each image, and the template images are represented by their hulls and sections – see Figure 6.23. The results of three phases of the algorithm are shown in Figure 6.24: the affine transformation of the template hulls at the end of the global pass, the corresponding transformation of the template sections at the beginning of the local pass, and the final piece-wise transformations of the template images. Table 6.4 presents the parameter ranges for the global search.

As one can see from the comparison of the results of the piece-wise transformation (Figure 6.24, rightmost column) with the original reference image in Figure 6.23 (left), the algorithm is able to find a fairly good mapping from the template images onto the reference image.

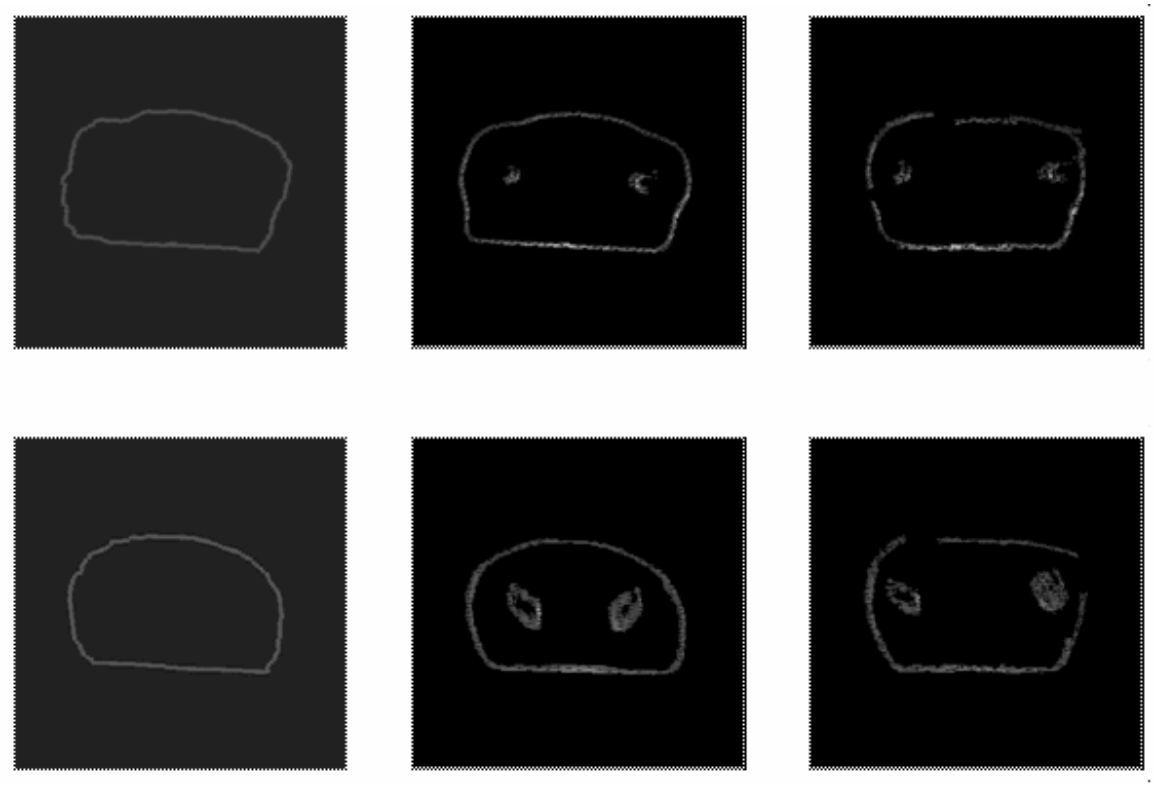

Figure 6.24: Results of three different phases of the algorithm for the template 1 (top row) and template 2 (bottom row): the global transformations of the hulls (left column); the initial transformations of the template sections (center column); and the final piece-wise transformations of the template sections (right column)

Parameter	***θ***	***DX***	***DY***	***SX***	***SY***	***SHX***	***SHY***
Lower limit	- 10°	0	0	0.5	0.5	- 0.3	- 0.3
Upper limit	+ 10°	360	360	1.0	1.0	+ 0.3	+ 0.3

Table 6.4: Parameter ranges for the global elastic registration of medical images

6.6. Summary

Chapter 6 presents a sample implementation of the optimization algorithm based on the range of models and algorithms proposed and discussed in Chapters 4 and 5. The algorithm extends and complements these models and algorithms in the following directions.

Optimization search is conducted entirely in the response space, which directly extracts the main shape features of an image, thus avoiding the unreliable direct comparison of image pixels on the one hand, and significantly reducing the amount of information that needs to be processed during fitness evaluation, on the other hand.

Multiple populations are used during the search, in order to deal with the applications that have to find mapping from multiple views onto the same reference image. Each of the multiple views is represented by the independent subpopulation. In order to reduce the dimensions of the search space and eliminate false solutions, the algorithm aligns all subpopulations on a sub-vector $\boldsymbol{V} = (\theta, SHX, SHY)$ of the general affine transformation, every time the new generation is created.

Since real-world imaging problems might include different views of the same object allowing for a significant distortion of the object image which cannot be represented by a single transformation vector, the optimization of a piece-wise affine transformation between the images subject to the mapping, with a hybrid EA model is introduced.

In order to increase the diversity of the population and provide a fair representation of all parameter ranges in the population, a frontal algorithm of creating new population is introduced, which keeps track of the usage of the parameter space during all operations of crossover, mutation, and local search.

In order to increase the robustness and accuracy of the image mapping, the algorithm conducts the search using multiple optimization objectives, in the form of different fitness functions. The algorithm processes the objectives in an alternating manner, switching between the fitness functions, at the different computational stages.

The computational model of an image is created using the following steps:

- Image local response is computed;
- histogram of the response is computed, and a threshold is applied to eliminate noise;
- image is divided into sections, such that each section has its own transformation vector;
- each image section is decomposed into a quadtree whose nodes participate in all computational operations;
- all image sections are organized in a hierarchical tree, and processed in a top-to-bottom manner;
- essential part (a hull) of the image is chosen, such that the hull's transformation can be defined as an elastic affine transformation; the hull is represented by its contour nodes and internal black nodes.

The algorithm works in two consecutive passes: during the first pass, global optimization is performed seeking for the optimum mapping of the image hull. During the second pass, the hull transformation is used as the initial approximation for the final piece-wise optimization of image sections.

During the second pass of the algorithm, image sections are organized in the hierarchical tree structure, where the top level is formed by the sections adjacent to the image hull. The algorithm processes the sections in a level-wise manner, starting from the top level. The piece-wise transformation of the parental level serves as the initial approximation for the optimization of the offspring level. The sections of the top level use the hull transformation as the starting point of the search.

In the second pass of the algorithm, the initial approximation for the piece-wise transformation is available from the global pass. The algorithm uses a principle of relaxation of parameter ranges when it generates the next population. The ranges are centered about the initial approximation, and are iteratively increased, until all potentially viable candidate solutions can be generated. This approach, on the one hand, favors the solutions that are closer to the initial approximation; on the other hand, it allows for the sections to undergo significant distortions during the mapping.

The algorithm intensively uses local optimization search. The fitter to date chromosomes are selected and clustered based on the matrix of their mutual Hamming distances. Each

cluster forms an individual DSM simplex; random mutation is used, if necessary, to complete the set of the simplex vertices.

The potential ability of the algorithm to solve complex image mapping problems involving significant image distortion, is illustrated on three sample imaging problems: global human body registration, 3-D object recognition in the 2-D space, and elastic registration of medical images.

Chapter 7: Summary and Conclusions

7.1. Objectives of the Research

This research applies Evolutionary algorithms (EAs) to the task of finding a proper mapping between geometrically distorted images, which arises in such applications as image registration, object recognition, and content-based image retrieval. The task can be formulated as an imaging optimization problem of minimizing the difference between the images. Continually growing complexity of imaging optimization problems and necessity to deal with changing dynamic and uncertain information put higher demands on the existing systems of processing visual information, and expose their limitations and frequent inefficiency, which makes researchers turn their attention to new methods that are capable of autonomous adjustment and self-adaptation to volatile inputs and tasks. One of the powerful and versatile among such methods is a family of Evolutionary algorithms (EAs) – heuristic-based global search and optimization methods originally based on the principles of biological evolution. The algorithms have three principal advantages over other computational methods:

- EAs use inherently parallel processing of information;
- EAs are efficient in conducting global search in a large multi-dimensional parameter space;
- EAs are highly tolerant to non-linearity, non-continuity, and irregularity of the objective function.

Analysis of the state-of-the-art in the Evolutionary algorithms, on the one hand, and its comparison with the range of the practical imaging optimization problems solved with EAs, on the other hand, shows significant gap between the two. While the current status in the EAs theory and methodology includes a broad variety of advanced models and techniques, the majority of the practical imaging applications still use the basic, classical models of the main representatives from the EAs family. The classical EA models have proven inefficient in many practical applications, including some of the problems related to image processing. However, the implementation of advanced methods that go beyond the classical models requires an in-depth understanding of the morphological development of Evolutionary algorithms, as well as a fairly comprehensive analysis of their recent trends and advances.

Challenges in solving complex real-world imaging optimization problems, and the shortcomings of the EA models currently used to solve these problems, suggest the overall objective of this research: it attempts to analyze the potential use of modern advances in Evolutionary computations for solving practical problems encountered in imaging optimization. In particular, the research attempts to achieve the following objectives:

- develop and analyze a range of models, algorithms, and techniques implementing advanced concepts of Evolutionary algorithms which can improve their computational performance in imaging optimization;

- design a fairly general approach based on a broader hybridization of EAs with a range of other advanced models, algorithms, and techniques, for solving a larger spectrum of imaging optimization problems.

7.2. Important Results of the Research

The research proposes the following main directions for improving the computational performance of Evolutionary algorithms in imaging optimization.

Hybridizing EAs with local optimization technique, in the particular form of a two-phase (random/direct) cyclic search algorithm, reduces the excessive computational cost of local search and refinement of the solution. The algorithm has the following distinctive features:

- using direct DSM search, as opposed to gradient-based methods of local search frequently used in the hybrid EA models, which allows one to reduce the total number of fitness evaluations and increase the robustness of the algorithm, in the case of possible irregularities of fitness function;
- adding randomness to the deterministic and algorithmically well defined DSM, which allows one to relocate the simplex in the local neighborhood in a more advantageous way, thus preventing the search from falling into a sub-optimum solution and reducing the total number of fitness evaluations;
- engaging partial, rather than the complete local search, in a cyclic manner, which allows one to reduce the number of wasted local evaluations, in the case of

discarding locally evaluated neighborhoods at a later stage of the global evolutionary search;

- building a tree structure, in order to keep track of the visited points in the local neighborhood, which prevents the search from re-visiting the discarded points.

Utilizing bookkeeping operation in the search space, in the form of mutation with memory and frontal algorithm of forming new population, assures its diversity and restores the fair and effective usage of the search space disrupted during the operations of crossover, mutation, and local search.

Comparative experiments on image sets undergoing a 4-dimensional affine mapping show a superior performance of the hybrid EA model that uses the proposed two-phase local search and mutation with memory, over the classical elitist EA model. For example, the classical EA model finds the minimum fitness values $F = 0.00464$ and $F = 0.00620$ after 391 and 77 generations, with the number of fitness evaluations $E = 44213$ and $E = 8731$, for the first and the second tested sets, respectively. The modified hybrid EA model is able to find better minimum fitness values $F = 0.00370$ and $F = 0.00246$ after 11 and 17 generations, with significantly smaller number of fitness evaluations $E = 3421$ and $E = 5467$, for the first and second tested sets, respectively.

Utilizing a concept of Image local response allows one to directly extract the main shape features of an image, reduce the computational cost of fitness evaluation, and provide an efficient image model for adaptive local search, reduction of parameter space, and multi-

sensor image fusion. Image local response has the following important properties that make it a valuable tool in imaging optimization with Evolutionary algorithms:

- response senses the distribution of pixel values in small localities across the image, thus providing a crude estimation of the local behavior of fitness function;
- response emphasizes the areas of the image that are most sensitive to the applied transformation, which makes it a convenient technique for extracting essential shape features of the image;
- since response is computed by mapping image onto itself, it is not particularly sensitive to the imagery type, which makes it a useful tool in multi-sensor fusion.

Comparative experiments on image sets undergoing a 5-dimensional affine mapping show the advantage of using image responses for selective fitness evaluation. For example, for two image sets, the full evaluation of the images requires $E = 9119$ and $E = 5236$ fitness evaluations, and provides the minimum fitness values $F = 0.00876$ and $F = 0.00317$, for the first and second sets, respectively. The use of the selective fitness evaluation based on image response mask requires significantly smaller number of evaluations, $E = 6088$ and $E = 2495$, and provides similar minimum fitness values $F = 0.01154$ and $F = 0.00248$, for the first and second sets, respectively.

The adaptive response-based mechanism used in local DSM search reduces the number of local fitness evaluations between 27.2% and 44.9%, without the loss of the quality of the optimum solution, on the tested image sets subject to a 5-dimensional affine mapping.

Cross correlation of image response matrices allows one to find the correct 5-dimensional mapping on a test set after 24 generations, when the regular algorithm stalls in a local minimum point. When applied to the second test set, this technique allows one to reduce the computational cost of the mapping to 7 generations, as opposed to 23 generations spent by the regular algorithm.

Experiments conducted on a sample test set, in the cases of 4- and 5-dimensional mapping problems, show that conducting the search in image response space, rather than in the actual visual space, allows one to successfully map images of different types, particularly, to map a wireframe object model onto the scene containing a solid object model.

Introducing an advanced image model reduces the amount of information needed for image processing during the search. The computational model of an image is created using the following steps:

- image response is computed;
- histogram of the response is computed, and a threshold is applied to eliminate noise;
- image is divided into sections, such that each section has its own transformation vector;
- each image section is decomposed into a quadtree whose nodes participate in all computational operations;

- image sections are organized in a hierarchical tree structure which is processed in a top-to-bottom manner;
- essential part (a hull) of the image shape is chosen, such that the hull's transformation can be defined as elastic affine transformation; the hull is represented by its contour nodes and internal black nodes.

Representing the sought image mapping in the form of a piece-wise affine transformation allows for significant mutual distortion of the compared images, such that the different image sections have their respective local affine transformations.

Organizing image sections as a tree structure and processing the tree in the hierarchical top-to-bottom manner, allows one to use the local transformations of the parental sections as initial approximations for the local transformations of the offspring sections.

Utilizing multiple aligned populations increases the coherence and robustness of the mapping when multiple image views have to be simultaneously processed. Each of the multiple views is represented by an independent subpopulation. In order to reduce the dimensions of the search space and eliminate false solutions, the algorithm aligns all subpopulations on the sub-vector $\boldsymbol{V} = (\theta, SHX, SHY)$ of the general affine transformation, every time the new generation is created. Here, θ is the rotation angle in the xy plane; and SHX and SHY are the shear factors along the x and y axes, respectively.

Utilizing multi-objective optimization, where the algorithm processes different fitness functions at the different computational stages, increases the efficiency and confidence of the search.

The algorithm works in two consecutive passes: during the first pass, global optimization is performed seeking for the optimum mapping of the image hull. During the second pass, the hull transformation is used as the initial approximation for the final piece-wise optimization of image sections, in the following way.

During the second pass of the algorithm, image sections are organized in a hierarchical tree structure, where the top level is formed by the sections adjacent to the image hull. The algorithm processes the sections in a level-wise manner, starting from the top level. The piece-wise transformation of the parental level serves as the initial approximation for the optimization of the offspring level. The sections of the top level use the hull transformation as the starting point of the search.

Moreover, in the second pass of the algorithm, the initial approximation for the piece-wise transformation of the top-level sections is available from the first, global pass. The algorithm uses a principle of relaxation of the parameter ranges, when it generates the next population for each section. The ranges are centered about the initial approximation; they are iteratively increased until all potentially viable candidate solutions can be generated. This approach, on the one hand, favors the solutions that are closer to the

initial approximation; on the other hand, it allows for the sections to undergo significant distortions during the mapping.

The algorithm intensively utilizes local optimization search. The fitter chromosomes are selected and clustered based on the matrix of their mutual Hamming distances. Each cluster forms an individual DSM simplex; random mutation is used, if necessary, to complete the set of the simplex vertices.

The potential ability of the algorithm to solve complex mapping problems involving significant image distortion is demonstrated on three sample imaging optimization problems: global human body registration, 3-D object recognition in the 2-D space, and elastic registration of medical images.

7.3. Potential Directions of Further Research

Analysis of the state-of-the-art of Evolutionary algorithms in imaging optimization and the results of this research allows one to identify some potentially efficient directions of further research toward the broad hybridization of Evolutionary algorithms with other methods, including the extension of both, application and methodology areas:

1. Application area - extension of the obtained theoretical and experimental results to interesting cases of other transformations, e.g., an important practical case of perspective transformation.
2. Further theoretical and experimental investigation of the proposed concept of Image local response including its properties; applicability to different imagery

and utilization in multi-sensor fusion; and the use for pre-processing and enhancing evolutionary operators.

3. Further development of the proposed two-phase cyclic local search algorithm including investigation of the statistical correlation, on the neighborhood level, between fitness function and image local response, in order to further increase the efficiency of the local search.
4. Further development of algorithms for managing the global search space during all operations of selection, crossover, mutation, and local search, based on the extension of the proposed frontal algorithm.
5. Development of more comprehensive criteria of terminating the evolutionary search, based on the use of statistical analysis and results of the memory management and control.

References

[1] Alander, J. T. Indexed Bibliography of Genetic Algorithms in Optics and Image Processing. University of Vaasa, Department of Information Technology and Production Economics, Report Series No 94-1-OPTICS, 2004.

[2] Anderson, R. W., Fogel, D. B., and Schultz, M. Other operators. In Evolutionary Computation 1: Basic Algorithms and Operators, pp. 308-325. Institute of Physics Publishing, Bristol, Philadelphia, 2000.

[3] Ankenbrandt, C. A., Buckles, B. P., and Petry, F. E. Scene Recognition Using Genetic Algorithms with Semantic Nets. Pattern Recognition Letters, Vol. 11, No. 4, pp. 231-304, 1990.

[4] Bäck, T. Evolutionary Algorithms in Theory and Practice. Oxford University Press, New York, 1996.

[5] Bäck, T. Introduction to Evolutionary Algorithms. In Evolutionary Computation 1: Basic Algorithms and Operators, pp. 59-63. Institute of Physics Publishing, Bristol, Philadelphia, 2000.

[6] Baker, J. E. Adaptive Selection Methods for Genetic Algorithms. In Proc. of the First International Conference on Genetic Algorithms and Their Applications, pp. 101-111. Lawrence Erlbaum Associates, Hillsdale, NJ, 1985.

[7] Baker, J. E. Reducing Bias and Inefficiency in the Selection Algorithm. In Proc. of the Second International Conference on Genetic Algorithms, pp. 14-21. Lawrence Erlbaum Associates, Hillsdale, NJ, 1987.

[8] Banzhaf, W., Nordin, P., Keller, R. E., and Francone, F. D. Genetic Programming: An Introduction. Morgan Kaufmann Publishers, San Fransisco, CA, 1998.

[9] Barton, G. Elements of Green's Functions and Propagation: Potentials, Diffusion, and Waves. Clarendon Press, Oxford, 1989.

[10] Besl, P. and McKay, N. A Method for Registration of 3-D Shapes. IEEE Trans. on Pattern Analysis and Machine Intelligence, Vol. 14, No. 2, pp. 239-256, 1992.

[11] Bornholdt, S. Genetic Algorithms. In Non-Standard Computation: Molecular Computation – Cellular Automata – Evolutionary Algorithms – Quantum Computers, pp. 59–94. Wiley-VCH, 1998.

[12] Brooks, R. R. and Iyengar, S. S. Multi-Sensor Fusion: Fundamentals and Applications with Software. Prentice Hall, New York, 1998.

[13] Buckles, B. P. and Petry, F. E. Cloud Identification Using Genetic Algorithms and Massively Parallel Computation. Final Report, Grant No. NAG 5-2216, Center for Intelligent and Knowledge-based Systems, Dept. of Computer Science, Tulane Univ., New Orleans, LA, June 18, 1996.

[14] Buzug, T. and Weese, J. Improving DSA Images with an Automatic Algorithm Based on Template Matching and an Entropy Measure. In Computer Assisted Radiology (CAR'96), pp. 145-150. Elsevier Science, Amsterdam, 1996.

[15] Calway, A. D., Knutsson, H., and Wilson, R. A Multiresolution Frequency Domain Algorithm for Fast Image Registration. In Proc. of The Third International Conference On Visiual Search, pp. 1-2. Nottingham, England, August 1992.

[16] Cantú-Paz, E. Efficient and Accurate Parallel Genetic Algorithms. Kluwer Academic Publishers, Boston, 2000.

[17] Chalermwat, P., El-Ghazawi, T., and LeMoigne, J. GA-Based Parallel Image Registration on Parallel Clusters. In IPPS/SPDP Workshops, pp. 257-265, 1999.

[18] Chen, Y. and Medioni, G. Object Modeling by Registration of Multiple Range Images. In Proc. IEEE Conf. on Robotics and Automation, Vol. 3, pp. 2724-2729, 1991.

[19] Chow, C. K., Tsui, H. T., and Lee, T. Surface Registration Using Dynamic Genetic Algorithm. Technical Report #001, CVIP Laboratory, Dept. of Electronic Engineering, The Chinese University of Hong Kong, 2001.

[20] Collignon, A., Maes, F., Delaere, D., Vandermeulen, D., Suetens, P., and Marchal, G. Automated Multimodality Image Registration Using Information Theory. In Information Processing in Medical Imaging, pp. 263-274. Kluwer Academic Publishers, Dordrecht, 1995.

[21] Davis, L. D. Adapting Operator Probabilities in Genetic Algorithms. In Proc. of the Third International Conference on Genetic Algorithms, pp. 61-69. Morgan Kaufmann Publishers, San Mateo, CA, 1989.

[22] De Bonet, J. S., Isbell, C. L., and Viola, P. MIMIC: Finding Optima by Estimating Probability Densities. Advances in Neural Information Processing Systems, No. 9, pp. 424-430, 1997.

[23] De Castro, E. and Morandi, C. Registration of Translated and Rotated Images Using Finite Fourier Transforms. IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 9, No. 5, pp. 700-702, 1987.

[24] De Jong, K.A. An Analysis of the Behavior of a Class of Genetic Adaptive Systems. PhD Dissertation, University of Michigan: Ann Arbor, 1975.

[25] De Jong, K. Evolutionary Computation: Recent Developments and Open Issues. In Evolutionary Algorithms in Engineering and Computer Science, pp. 43-54. John Wiley & Sons, 1999.

[26] Deb, K. Multi-Objective Optimization using Evolutionary Algorithms. John Wiley & Sons, 2001.

[27] Deb, K. and Goldberg, D. E. An Investigation of Niche and Species Formation in Genetic Function Optimization. In Proc. of the Third International Conference on Genetic Algorithms, pp. 42-50. Morgan Kaufmann Publishers, San Mateo, CA, 1989.

[28] Delorme, A. and Thorpe, S. Face Identification Using One Spike Per Neurone: Resistance to Image Degradation. Neural Networks, Special Issue, No. 14, pp. 795-808, 2001.

[29] Delorme, A., Gautrais, J., Van Rullen, R., Thorpe, S. SpikeNET: A Simulator for Modeling Large Networks of Integrate and Fire Neurons. Neurocomputing, No. 26-27, pp. 989-996, 1999.

[30] Dzemyda, G., Saltenis, V., and Zhilinskas, A. Stochastic and Global Optimization. Kluwer Academic Publishers, 2002.

[31] Eiben, A. E., Raué, P.-E., and Ruttkay, Z. Genetic Algorithms with Multi-Parent Recombination. In Parallel Problem Solving from Nature – PPSN III, Lecture Notes in Computer Science, Vol. 866, pp. 77-87. Springer-Verlag, Berlin, 1994.

[32] Eshelman, L. J. and Schaffer, J. D. Real-Coded Genetic Algorithms and Interval-Schemata. In Foundations of Genetic Algorithms 2 (FOGA-2), pp. 187-202. Morgan Kaufmann Publishers, San Mateo, CA, 1993.

[33] Fagin, R. and Stockmeyer, L. Relaxing The Triangle Inequality in Pattern Matching. International Journal of Computer Vision, Vol. 28, No. 3, pp. 219–231, 1998.

[34] Fitzpatrick, J. M. and Grefenstette, J. J. Genetic Algorithms in Noisy Environments. Machine Learning, No. 3, pp. 101-120, 1988.

[35] Fitzpatrick, J. M., Grefenstette, J. J., and Van Gucht, D. Image Registration by Genetic Search. In IEEE Southeastcon ’84, Louisville, KY, 1984.

[36] Fogel, D. B. Some Recent Important Foundational Results in Evolutionary Computation. In Evolutionary Algorithms in Engineering and Computer Science, pp. 55-71. John Wiley & Sons, 1999.

[37] Fogel, D.B. Introduction to Evolutionary Computation. In Evolutionary Computation 1: Basic Algorithms and Operators, pp. 1-3. Institute of Physics Publishing, Bristol, Philadelphia, 2000.

[38] Fogel, D. B. Evolutionary Computation: Toward a New Philosophy of Machine Intelligence. IEEE Press, New York, 2000.

[39] Fogel, D. B. and Ghozeil, A. A Note on Representations and Variation Operators. IEEE Trans. on Evolutionary Computation, Vol. 1, No. 2, pp. 159-161, 1997.

[40] Fogel, L. J. On the Organization of Intellect. PhD Dissertation, University of California: Los Angeles, 1964.

[41] Fogel, L. J., Owens, A. J., and Walsh, M. J. Artificial Intelligence through Simulated Evolution. Wiley, New York, 1966.

[42] Fukuda, T., Kubota, N., and Shimojima, K. Virus-Evolutionary Genetic Algorithm and Its Application to Traveling Salesman Problem. In Evolutionary Computation: Theory and Applications, pp. 235-255. World Scientific Publishing, 1999.

[43] Gautrais, J. and Thorpe, S. Rate Coding Versus Temporal Order Coding: A Theoretical Approach. BioSystems, No. 48, pp. 57-65, 1998.

[44] Gen, M. and Cheng, R. Genetic Algorithms and Engineering Optimization. Wiley, New York, 2000.

[45] Gen, M. and Liu, B. A Genetic Algorithm for Nonlinear Goal Programming. Evolutionary Optimization, Vol. 1, No. 1, pp. 65-76, 1999.

[46] Glover, F. and Laguna, M. Tabu Search. Kluwer Academic Publishers, 1997.

[47] Goldberg, D. E. Simple Genetic Algorithms and the Minimal Deceptive Problem. In Genetic Algorithms and Simulated Annealing, pp. 74-88. Morgan Kaufmann Publishers, Los Altos, CA, 1987.

[48] Goldberg, D. E. Genetic Algorithms in Search, Optimization, and Machine Learning. Addison-Wesley, Reading, MA, 1989.

[49] Goldberg, D. E. The Design of Innovation: Lessons from and for Competent Genetic Algorithms. Kluwer Academic Publishers, Boston, 2002.

[50] Goldberg, D. E. and Deb, K. A Comparative Analysis of Selection Schemes Used in Genetic Algorithms. In Foundations of Genetic Algorithms, pp. 69-93. Morgan Kaufmann Publishers, San Mateo, CA, 1991.

[51] Goldberg, D. E. and Richardson, J. Genetic Algorithms with Sharing for Multimodal Function Optimization. In Proc. of the Second International Conference on Genetic Algorithms, pp. 41-49. Lawrence Erlbaum Associates, Cambridge, MA, 1987.

[52] Goldberg, D. E., Deb, K., and Clark, J. H. Genetic Algorithms, Noise, and the Sizing of Populations. Complex Systems, Vol. 6, No. 4, pp. 333-362, 1992.

[53] Goldberg, D. E., Deb, K., and Thierens, D. Toward a Better Understanding of Mixing in Genetic Algorithms. Journal of the Society of Instrument and Control Engineers, Vol. 32, No. 1, pp. 10-16, 1993.

[54] Goldberg, D. E., Deb, K., Kargupta, H., and Harik, G. Rapid, Accurate Optimization of Difficult Problems Using Fast Messy Genetic Algorithms. In Proc. of the Fifth International Conference on Genetic Algorithms, pp. 56-64. Morgan Kaufmann, San Mateo, CA, 1993.

[55] Grefenstette, J. J. and Fitzpatrick, J. M. Genetic Search with Approximate Function Evaluations. In Proc. of the First International Conference on Genetic Algorithms and Their Applications, pp. 112-120. Lawrence Erlbaum, Pittsburgh, PA, 1985.

[56] Guerra-Salcedo, C. and Whitley, D. Feature Selection Mechanisms for Ensemble Creation: A Genetic Search Perspective. In Data Mining with Evolutionary Algorithms:

Research Directions: Papers from the AAAI Workshop, Technical Report WS-99-06, pp. 13-17. AAAI Press, 1999.

[57] Hall, D. L. and Llinas, J. An Introduction to Multisensor Data Fusion. Proc. of the IEEE, Vol. 85, No. 1, pp. 6-23, 1997.

[58] Harik, G. Learning gene linkage to effectively solve problems of bounded difficulty using genetic algorithms. PhD Dissertation, University of Michigan: Ann Arbor, 1997.

[59] Harik, G., Lobo, F. G., and Goldberg, D. E. The Compact Genetic Algorithm. In Proc. of the 1998 IEEE Conference on Evolutionary Computation, pp. 523-528, 1998.

[60] Hart, W. E. and Belew, R. K. Optimization with Genetic Algorithm Hybrids That Use Local Search. In Adaptive Individuals in Evolving Populations: Models and Algorithms, 1996: Proc. of Santa Fe Institute Studies in the Sciences of Complexity, Vol. 26, pp. 483-496. Addison-Wesley, Reading, MA, 1996.

[61] Haupt, R. L. and Haupt, S. E. Practical Genetic Algorithms. John Wiley & Sons, New York, 1998.

[62] Haykin, S. Communication Systems. John Wiley & Sons, Inc., 1994.

[63] Herrera, F., Lozano, M., and Verdegay, J. L. Tackling Fuzzy Genetic Algorithms. In Genetic Algorithms in Engineering and Computer Science, pp. 167-189. John Wiley & Sons, 1995.

[64] Hesser, J. and Männer, R. Towards an Optimal Mutation Probability for Genetic Algorithms. In Proc. of the 1st Conference on Parallel Problem Solving from Nature, Lecture Notes in Computer Science, Vol. 496, pp. 23-32. Springer, Berlin, 1991.

[65] Hill, A. and Taylor, C. J. Model-Based Image Interpretation Using Genetic Algorithms. Image and Vision Computing, Vol. 10, No.5, pp. 295-300, 1992.

[66] Holland, J. H. Adaptation in Natural and Artificial Systems. University of Michigan Press: Ann Arbor, 1975; MIT Press, 1992 (2nd edition).

[67] Horn, J., Nafpliotis, N., and Goldberg, D. A Niched Pareto Genetic Algorithm for Multiobjective Optimization. In Proc. of the 1994 IEEE Conference on Evolutionary Computation, pp. 82-87. IEEE Press, Piscataway, NJ, 1994.

[68] Hu, Changbo, Li, Qingfeng Yu Yi, and Ma, Songde. Extraction of Parametric Human Model for Posture Recognition Using Genetic Algorithm. In Proc. of the Fourth IEEE International Conference on Automatic Face and Gesture Recognition, pp. 518—523, 2000.

[69] Ibáñez, L., Schroeder, W., Ng, L., and Cates, J. The ITK Software Guide. Insight Software Consortium, 2003.

[70] Image ID# EL-1996-00037. NASA Langley Research Center – Multimedia Repository, <http://lisar.larc.nasa.gov>.

[71] Image ID# line0628. NOAA Central Library, America's Coastlines Collection, <http://www.photolib.noaa.gov/coastline/>.

[72] Image ID# tank4 3ds. Princeton Shape Retrieval and Analysis Group, 3D Model Search Engine, <http://shape.cs.princeton.edu/search.html>.

[73] Image ID# theb3002. NOAA Central Library, Historic C&GS Collection, <http://www.photolib.noaa.gov/coastline/>.

[74] Image ID# town04. CMU/VASC Image Database, <http://vasc.ri.cmu.edu/idb/html/motion/index.html>.

[75] Image ID# town06. CMU/VASC Image Database, <http://vasc.ri.cmu.edu/idb/html/motion/index.html>.

[76] Image ID# wea00132. NOAA Central Library, National Weather Service Historical Image Collection, < http://www.photolib.noaa.gov/historic/nws/>.

[77] Image ID# wea00151. NOAA Central Library, National Weather Service Historical Image Collection, < http://www.photolib.noaa.gov/historic/nws/>.

[78] Image Processing Toolbox User's Guide. The MathWorks, Inc., 1993-2006.

[79] Iqbal, Q. and Aggarwal, J. K. Lower-Level and Higher-Level Approaches to Content-Based Image Retrieval. In Proceedings of the IEEE South West Symposium on Image Analysis and Interpretation, pp. 197-201. Austin, TX, 2000.

[80] Joines, J. A. and Kay, M. G. Utilizing Hybrid Genetic Algorithms. In Evolutionary Optimization, pp. 199-228. Kluwer Academic Publishers, Boston, MA, 2002.

[81] Jones, S. M. and Boyer, A. L. Investigation of an FFT-Based Correlation Technique for Verification of Radiation Treatment Setup. Medical Physics, Vol. 18, No. 6, pp. 1116-1125, 1991.

[82] Junck, L., Moen, J. G., Hutchins, G. D., Brown, M. B., and Kuhl, D. E. Correlation Methods for the Centering, Rotation, and Alignment of Functional Brain Images. Journal of Nuclear Medicine, Vol. 31, No. 7, pp. 1220-1226, 1990.

[83] Kargupta, H. Gene Expression: The Missing Link in Evolutionary Computation. In Genetic Algorithms and Evolution Strategies in Engineering and Computer Science. Recent Advances and Industrial Applications, pp. 59-83. John Wiley & Sons, 1998.

[84] Kim, Y., Street, W. N., and Menczer, F. Feature Selection in Unsupervised Learning via Evolutionary Search. In Proc. 6th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD-00), pp. 365-369, 2000.

[85] Kitano, H. Designing Neural Networks Using Genetic Algorithms with Graph Generation System. Complex Systems, Vol. 4, No. 4, pp. 461-476, 1990.

[86] Knjazew, D. OmeGA: A Competitive Genetic Algorithm for Solving Permutation and Scheduling Problems. Kluwer Academic Publishers, 2002.

[87] Kolda, T. G., Lewis, R. M., and Torczon, V. Optimization by Direct Search: New Perspectives on Some Classical and Modern Methods**.** SIAM Review, Vol. 45, No. 3, pp. 385-482, 2003.

[88] Koza, J. R. Hierarchical Genetic Algorithms Operating on Populations of Computer Programs. In Proc. of the 11th International Joint Conference on Artificial Intelligence, pp. 768-774. Morgan Kaufman Publishers, San Mateo, CA, 1989.

[89] Koza, J. R. Genetic Programming: On the Programming of Computers by Means of Natural Selection. MIT Press, Cambridge, MA, 1992.

[90] Krink, T., Mayoh, B. H., and Michalewicz, Z. A PATCHWORK Model for Evolutionary Algorithms with Structured and Variable Size Populations. In Proc. of the Genetic and Evolutionary Computation Conference, pp. 1321-1328. Orlando, FL, 1999.

[91] Krishnakumar, K., Narayanaswamy, S., and Garg, S. Solving Large Parameter Optimization Problems Using a Genetic Algorithm with Stochastic Coding. In Genetic Algorithms in Engineering and Computer Science, pp. 287-303. John Wiley & Sons, Chichester; New York, 1995.

[92] Krüger, S. and Calway, A. Image Registration Using Multiresolution Frequency Domain Correlation. In British Machine Vision Conference, pp. 316-325, Sep. 1998.

[93] Kudo, M. and Skalansky, J. Comparison of Algorithms That Select Features for Pattern Classifiers. Pattern Recognition, Vol. 33, No. 1, pp. 25-41, 2000.

[94] Larañaga, P. A Review on Estimation of Distribution Algorithms. In Estimation of Distribution Algorithms: A New Tool for Evolutionary Computation, pp. 57-100. Kluwer Academic Publishers, 2002.

[95] Leardi, R., Boggia, R. and Terrile, M. Genetic Algorithms as a Strategy for Feature Selection. Journal of Chemometrics, Vol. 6, No. 5, pp. 267-281, 1992.

[96] Learning MATLAB. MathWorks, 2001.

[97] Lee, C. C. Fuzzy Logic in Control Systems: Fuzzy Logic Controller, Part I and II. IEEE Trans. on Systems, Man and Cybernetics, Vol. 20, No. 2, pp. 404-435, 1990.

[98] Lee, M. A. and Takagi, H. Dynamic Control of Genetic Algorithms Using Fuzzy Logic Techniques. In Proc. of the Fifth International Conference on Genetic Algorithms, pp. 76-83. Morgan Kaufmann Publishers, San Mateo, CA, 1993.

[99] Locatelli, M. and Schoen, F. Random Linkage: A Family of Acceptance/Rejection Algorithms for Global Optimization. Mathematical Programming, Vol. 85, No.2, pp. 379-396, 1999.

[100] Maes, F., Collignon, A., Vandermeulen, D., Marchal, G., and Suetens, P. Multimodality Image Registration by Maximization of Mutual Information. IEEE Trans. on Medical Imaging, Vol. 16, No. 2, pp. 187-198, 1997.

[101] Man, K. F., Tang, K. S., and Kwong, S. Genetic Algorithms: Concepts and Design. Springer, 1999.

[102] Mandava, V. R., Fitzpatrick, J. M., and Pickens, D. R., III. Adaptive Search Space Scaling in Digital Image Registration. IEEE Trans. on Medical Imaging, Vol. 8, No. 3, pp. 251-262, 1989.

[103] Martin-Bautista, M. J. and Vila, M. A. A Survey of Genetic Feature Selection in Mining Issues. In Proc. Congress on Evolutionary Computation (CEC-99), pp. 1314-1321. Washington, D.C., 1999.

[104] McAulay, A. D. and Oh, J. C. Image Learning Classifier System Using Genetic Algorithms. In Proc. of the IEEE 1989 Aerospace and Electronics Conference NAECON, Vol. 2, Dayton, OH, 22-26 May, 1989, pp. 705-71. IEEE, New York, NY, 1989.

[105] Menczer, F., Degeratu, M., and Street, W. Efficient and Scalable Pareto Optimization by Evolutionary Local Selection Algorithms. Evolutionary Computation, Vol. 8, No. 2, pp. 223-247, 2000.

[106] Meyer, L. and Feng, X. A Fuzzy Stop Criterion for Genetic Algorithms Using Performance Estimation. In Proc. of the Third IEEE Conference on Fuzzy Systems, pp. 1990-1995. IEEE Press, Orlando, FL, 1994.

[107] Michalewicz, Z. Genetic Algorithms + Data Structures = Evolution Programs. Springer-Verlag, Berlin, 1992; 1996 (3rd edition).

[108] Michalewicz, Z. The Significance of the Evaluation Function in Evolutionary Algorithms. In Evolutionary Algorithms, pp. 151-166. Springer-Verlag, New York, 1999.

[109] Michalewicz, Z. and Fogel, D. B. How to Solve It: Modern Heuristics. Springer, Berlin; New York, 2000.

[110] Michalewicz, Z. and Janikow, C. Z. Handling Constraints in Genetic Algorithms. In Proc. of the Fourth International Conference on Genetic Algorithms, pp. 151-157. Morgan Kaufmann Publishers, San Mateo, CA, 1991.

[111] Mitchell, M. An Introduction to Genetic Algorithms. MIT Press, Cambridge, MA, 1996.

[112] Moscato, P. and Norman, A. A Memetic Approach for the Traveling Salesman Problem: Implementation of a Computational Ecology for Combinatorial Optimization on Message-Passing Systems. In Proc. of the International Conference on Parallel Computing and Transputer Applications, pp. 187-194. IOS Press, Amsterdam, 1992.

[113] Muchnik, I. and Mottl, V. Bellman Functions on Trees for Segmentation, Generalized Smoothing, Matching and Multi-Alignment in Massive Data Sets. DIMACS Technical Report 98-15, February 1998.

[114] Mühlenbein, H. The Equation for Response to Selection and Its Use for Prediction. Evolutionary Computation, Vol. 5, No. 3, pp. 303-346, 1998.

[115] Murata, T., Ishibuchi, H., and Tanoka, H. Multiobjective Genetic Algorithm and Its Application to Flowshop Scheduling. Computers and Industrial Engineering, Vol. 30, No. 4, pp. 957-968, 1996.

[116] Nelder, J. A. and Mead, R. A Simplex Method for Function Minimization. Computer Journal, No. 7, pp. 308-313, 1965.

[117] Niblack, W., Barber, R., Equitz, W., Flickner, M., Glasman, E., Petkovic, D., Yanker, P., Faloutsos, C. and Taubin, G. The QBIC Project: Querying Images by Content Using Color, Texture, and Shape. In Storage and Retrieval for Image and Video Databases. Proc. SPIE, Vol. 1908, pp. 173-187, April 1993.

[118] Papoulis, A. Signal Analysis. McGraw-Hill Book Company, 1977.

[119] Papoulis, A. Probability, Random Variables, and Stochastic Processes. McGraw Hill, 1991.

[120] Pelikan, M., Goldberg, D. E., and Cantú-Paz, E. BOA: The Bayesian Optimization Algorithm. In Proc. of the Genetic and Evolutionary Computation Conference, Vol. 1, pp. 525-532, 1999.

[121] Pitas, I. Digital Image Processing Algorithms and Applications. John Wiley & Sons, Inc., 2000.

[122] Pluim, Josien P. W., Maintz, J. B. Antoine, and Viergever, Max A. Interpolation Artifacts in Mutual Information-Based Image Registration. Computer Vision and Image Understanding, Vol. 77, No. 2, pp. 211-232, 2000.

[123] Prabhu, D., Buckles, B. P., and Petry, F. E. Genetic Algorithms for Scene Interpretation from Prototypical Semantic Description. International Journal of Intelligent Systems, Vol. 15, No. 10, pp. 901-918, 2000.

[124] Press, W. H., Teukolsky, S. A., Vetterling, W. T., and Flannery, B. P. Numerical Recipes in C. Cambridge University Press. Cambridge, UK, 1988.

[125] Prügel-Bennett, A. The Mixing Rate of Different Crossover Operators. In Foundations of Genetic Algorithms 6 (FOGA-6), pp. 261-274. Morgan Kaufmann Publishers, 2001.

[126] Radcliffe, N. and Surry, P. Formal Memetic Algorithms. In Evolutionary Computing, pp. 1-16. Springer-Verlag, Berlin, 1994.

[127] Radcliffe, T., Rajapakshe, R., and Shalev, S. Pseudocorrelation: A Fast, Robust, Absolute, Grey Level Image Alignment Algorithm. In Medical Imaging 1993: Image Processing. Proc. of SPIE, Vol. 1898, pp. 134-145. SPIE Press, Bellingham, WA, 1993.

[128] Raich, A.M. and Ghaboussi, J. Applying the Implicit Redundant Representation Genetic Algorithm in an Unstructured Problem Domain. In The Practical Handbook of Genetic Algorithms: Applications, pp. 295-340. Chapman & Hall/CRC, 2001.

[129] Rechenberg, I. Cybernetic: Solution Path of an Experimental Problem. Ministry of Aviation, Royal Aircraft Establishment, UK, 1965.

[130] Rechenberg, I. EvolutionsStrategie: Optimierung Technischer Systems nach Prinzipien der Biologischen Evolution. Frommann-Holzboog Verlag, Stuttgart, 1973.

[131] Reeves, C. R. Direct Statistical Estimation of GA Landscape Properties. In Foundations of Genetic Algorithms 6 (FOGA-6), pp. 91-107. Morgan Kaufmann Publishers, 2001.

[132] Reeves, C. R. and Yamada, T. Embedded Path Tracing and Neighborhood Search Techniques in Genetic Algorithms. In Evolutionary Algorithms in Engineering and Computer Science, pp. 95-111. John Wiley & Sons, 1999.

[133] Reynolds, R. G. An Introduction to Cultural Algorithms. In Proc. of the Third Annual Conference on Evolutionary Programming, pp. 131-139. World Scientific, River Edge, NJ, 1994.

[134] Rosenfeld, A. and Kak, A. C. Digital Picture Processing. Academic Press, New York, 1976.

[135] Rouet, J.-M. Robust 3-D Elastic Multimodality Image Registration through Genetic Algorithms. Internal Report, Thésard Département ITI, May 27, 1999.

[136] Rusinkiewicz, Szymon and Levoy, Marc. Efficient Variants of the ICP Algorithm. In Third International Conference on 3D Digital Imaging and Modeling (3DIM), pp. 145-152, 2001.

[137] Sakawa, M. Genetic Algorithms and Fuzzy Multiobjective Optimization. Kluwer Academic Publishers, 2002.

[138] Sarma, J. and De Jong, K. An Analysis of Local Selection Algorithms in a Spatially Structured Evolutionary Algorithm. In Proc. of the Seventh International Conference on Genetic Algorithms, pp. 181-187. Morgan Kaufmann, San Francisco, CA, 1997.

[139] Sarma, J. and De Jong, K. An Analysis of the Effects of Neighborhood Size and Shape on Local Selection Algorithms. In Parallel Problem Solving with Nature, pp. 236-244. Springer-Verlag, Berlin, 1996.

[140] Schaffer, J. Muliple Objective Optimization with Vector Evaluated Genetic Algorithms. In Proc. of the First International Conference on Genetic Algorithms, pp. 93-100. Lawrence Erlbaum Associates, Hillsdale, NJ, 1985.

[141] Schlosser, S. G., Trenkle, J. M., and Vogt, R. C. Feature Set Optimization for the Recognition of Arabic Characters Using Genetic Algorithms. In Proc. of the 21st Applied Imagery Pattern Recognition Workshop. Proc. SPIE, Vol. 1838, pp. 64-73. Washington, D.C., 1992.

[142] Schwefel, H.-P. Kybernetische Evolution als Strategie der Experimentellen Forshung in der Strömungstechnik. Diploma Thesis, Technical University of Berlin, 1965.

[143] Schwefel, H.-P. Numerical Optimization of Computer Models. Wiley, Chichester, 1981.

[144] Schwefel, H.-P. and Bäck, T. Artificial Evolution: How and Why? In Genetic Algorithms and Evolution Strategies in Engineering and Computer Science. Recent

Advances and Industrial Applications, pp. 1-19. John Wiley & Sons, Chichester, NY, 1998.

[145] Seul, M., O'Gorman, L., Sammon, M. J. Practical Algorithms for Image Analysis: Description, Examples, and Code. Cambridge University Press, Cambridge, UK, 2000.

[146] Shapiro, B. and Navetta, J. A Massively Parallel Genetic Algorithm for RNA Secondary Structure Prediction. The Journal of Supercomputing, Vol. 8, No. 3, pp. 195-207, 1994.

[147] Shekarforoush, H., Berthod, M., and Zerubia, J. Subpixel Image Registration by Estimating the Polyphase Decomposition of Cross Power Spectrum. In Proc. of IEEE Conference on Computer Vision and Pattern Recognition, pp. 532-537. San Francisco, CA, June 1996.

[148] Shen, Shuhan and Chen, Weirong. Probability Evolutionary Algorithm Based Human Body Tracking. In EvoWorkshops 2006, Lecture Notes in Computer Science, Vol. 3907, pp. 525–529. Springer, 2006.

[149] Siedlecki, W. and Sklansky, J. A Note on Genetic Algorithms for Large-Scale Feature Selection. Pattern Recognition Letters, Vol. 10, No. 5, pp. 335-347, 1989.

[150] Spears, W. M. and De Jong, K. A. On the Virtues of Parameterized Uniform Crossover. In Proc. of the Fourth International Conference on Genetic Algorithms, pp. 230-236. Morgan Kaufmann Publishers, San Mateo, CA, 1991.

[151] SpikeNet Technology Home Page, <http://www.spikenet-technology.com>.

[152] Studholme, C., Hills, D.L.G., and Hawkes, D.J. An Overlap Invariant Entropy Measure of 3D Medical Image Alignment. Pattern Recognition, Vol. 32, pp. 71-86, 1999.

[153] Syswerda, G. Uniform Crossover in Genetic Algorithms. In Proc. of the Third International Conference on Genetic Algorithms, pp. 2-9. Morgan Kaufmann Publishers, San Mateo, CA, 1989.

[154] Tanese, R. Parallel Genetic Algorithm for a Hypercube. In Proc. of the Second International Conference on Genetic Algorithms, pp. 177-183. Lawrence Erlbaum Associates, Hillsdale, NJ, 1987.

[155] Tanese, R. Distributed genetic algorithms for function optimization. PhD Dissertation, University of Michigan: Ann Arbor, 1989.

[156] Tang, J. and Wang, D. An Interactive Approach Based on a GA for a Type of Quadratic Programming Problem with Fuzzy Objective and Resources. Computers and Operations Research, Vol. 24, No. 5, pp. 413-422, 1997.

[157] Thorpe, S. Ultra-Rapid Scene Categorisation with a Wave of Spikes. In Biologically Motivated Computer Vision. Lecture Notes in Computer Science, Vol. 2525, pp. 1-15. Springer-Verlag, Berlin, 2002.

[158] Thorpe, S., Delorme, A., Van Rullen, R. Spike-Based Strategies for Rapid Processing. Neural Networks, Special Issue, No. 14, pp. 715-725, 2001.

[159] Trenkle, J. M., Schlosser, S. G., and Vogt, R.C. Morphological Feature-Set Optimization Using the Genetic Algorithm. In Proc. of Image Algebra and Morphological Processing II. Proc. SPIE, Vol. 1568, pp. 212-223. San Diego, CA, 1991.

[160] Turton, B., Arslan, T., and Horrocks, D. A Hardware Architecture for a Parallel Genetic Algorithm for Image Registration. In Proc. of the IEE Colloquium on Genetic Algorithms in Image Processing and Vision, pp. 11/1--11/6, October 1994.

[161] Vafaie, H. and De Jong, K. Robust Feature Selection Algorithms. In Proc. 1993 IEEE International Conference on Tools with AI, pp. 356-363. Boston, MS, USA, 1993.

[162] Van den Elsen, P. A., Pol, E.-J. D., Sumanaweera, T. S., Hemler, P.F., Napel, S., and Adler, J. R. Grey Value Correlation Techniques Used for Automatic Matching of CT and MR Brain and Spine Images. In Visualization in Biomedical Computing. Proc. of SPIE, Vol. 2359, pp. 227-237. SPIE Press, Bellingham, WA, 1994.

[163] Van Rullen, R. and Thorpe, S. Rate Coding Versus Temporal Order Coding: What the Retinal Ganglion Cells Tell the Visual Cortex. Neural Computation, No. 13, pp. 1255-1283, 2001.

[164] Van Rullen, R. and Thorpe, S. Surfing a Spike Wave Down the Ventral Stream. Vision Research, No. 42, pp. 2593-2615, 2002.

[165] Van Rullen, R., Gautrais, J., Delorme, A., and Thorpe, S. Face Processing Using One Spike Per Neurone. BioSystems, No. 48, pp. 229-239, 1998.

[166] Viola, P. Alignment by Maximization of Mutual Information. PhD Thesis, MIT, 1995.

[167] Viola, P. and Wells III, W. M. Alignment by Maximization of Mutual Information. In Proc. of the Fifth International Conference on Computer Vision, pp. 16-23. Los Alamos, CA, June 1995.

[168] Viola, Paul and Wells III, William M. Alignment by Maximization of Mutual Information. International Journal of Computer Vision, Vol. 24, No. 2, pp. 137–154, 1997.

[169] Voigt, H. M. Fuzzy Evolutionary Algorithms**.** Technical Report 92-038, International Computer Science Institute (ICSI), Berkeley, CA, 1992.

[170] Vose, M. D. The Simple Genetic Algorithm: Foundations and Theory. MIT Press, Cambridge, 1999.

[171] Vose, M. D. and Liepins, G. E. Punctuated Equilibria in Genetic Search. Complex Systems, Vol. 5, No. 1, pp. 31-44, 1991.

[172] Wang, J.-Z., Reinstein, L. E., Hanley, J., and Meek, A. G. Investigation of a Phase-Only Correlation Technique for Anatomical Alignment of Portal Images in Radiation Therapy. Physics in Medicine and Biology, Vol. 41, No. 6, pp. 1045-1058, 1996.

[173] Wells III, W. M., Viola, P., Atsumi, H., Nakajima, S., Kikinis, R. Multi-Modal Volume Registration by Maximization of Mutual Information. Medical Image Analysis, Vol. 1, No. 1, 1996, pp. 35-52.

[174] Welstead, S. T. Fractal and Wavelet Image Compression Techniques. SPIE Press, USA, 1999.

[175] Wolpert, D. H. and Macready, W. G. No Free Lunch Theorems for Optimization. IEEE Trans. on Evolutionary Computation, Vol. 1, No. 1, pp. 67-82, 1997.

[176] Wright, M. H. Direct Search Methods: Once Scorned, Now Respectable. In Numerical Analysis, 1995: Proc. of the 1995 Dundee Biennial Conference in Numerical Analysis, pp. 191-208. Addison Wesley Longman, Harlow, UK, 1996.

[177] Yang, J. and Honavar, V. Feature Subset Selection Using a Genetic Algorithm. Technical Report TR97-02, Dept. of Computer Science, Iowa State University, 1997.

[178] Yang, J. and Honavar, V. Feature Subset Selection Using a Genetic Algorithm. In Feature Extraction, Construction and Selection: A Data Mining Perspective, pp. 117-136. Kluwer, 1998.

[179] Yao, X. Introduction. In Evolutionary Computation: Theory and Applications, pp. 1-36. World Scientific Publishing, 1999.

[180] Zhang, L., Xu, W., and Chang, C. Genetic Algorithm for Affine Point Pattern Matching. Pattern Recognition Letters, Vol. 24, No. 1-3, pp. 9–19, 2003.

[181] Zhao, Jianhui and Li, Ling. Human Motion Reconstruction from Monocular Images Using Genetic Algorithms. Comp. Anim. Virtual Worlds, No. 15, pp. 407–414, 2004.