A Genetic algorithm for the detection of 2D geometric primitives in images

Evelyne Lutton, P. Martinez

INRIA

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

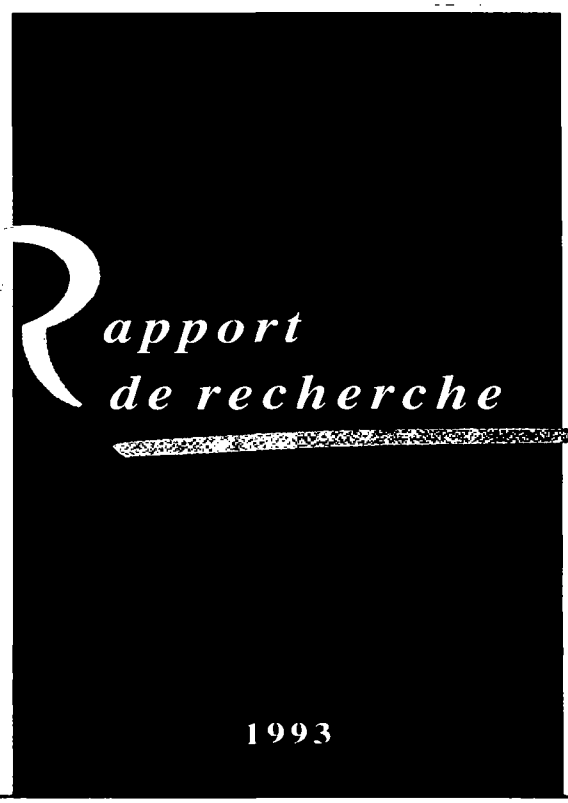
A Genetic Algorithm for the Detection of 2D Geometric Primitives in Images

Evelyne LUTTON
Patrice MARTINEZ

N° 2110
Novembre 1993

PROGRAMME 4

Robotique, image
et
vision



Un Algorithme Génétique pour la détection de primitives géométriques bidimensionnelles dans les images

Evelyne LUTTON, Patrice MARTINEZ
INRIA - Rocquencourt
B.P. 105, 78153 LE CHESNAY Cedex, France
Tel : 33 1 39 63 55 23 - Fax : 33 1 39 63 53 30
email : evelyne.lutton@inria.fr

November 10, 1993

Abstract

Nous étudions l'emploi des algorithmes génétiques dans le cadre de l'extraction de primitives (segments, cercles, quadrilatères, etc ...) dans des images. Cette approche est complémentaire de la transformée de Hough, dans le sens où les algorithmes génétiquess se révèlent efficaces là où la Transformée de Hough devient trop complexe et trop gourmande en espace mémoire, c'est-à-dire dans les cas où l'on recherche des primitives ayant plus de 3 ou 4 paramètres.

En effet, les algorithmes génétiques peuvent être employés en tant qu'algorithmes d'optimisation stochastiques. Cet outil d'optimisation peut se montrer très lent, mais se révèle efficace dans les cas où les fonctions à optimiser sont très irrégulières et de forte dimmensionnalités. La philosophie de la méthode que nous présentons est donc très similaire à celle de la transformée de Hough, qui est de rechercher un optimum dans un espace de paramètres. Cependant, nous verrons que les implantations algorithmiques diffèrent.

Cette approche de l'extraction de primitives par algorithmes génétiques n'est pas une idée nouvelle : nous avons repris et amélioré une technique originale proposée par Roth et Levine en 1992. Nous pouvons résumer notre apport sur cette technique en trois points principaux:

- nous avons utilisé des images de distances pour "adoucir" la fonction à optimiser,
- pour détecter plusieurs primitives à la fois, nous avons implanté et amélioré une technique de partage de la population (technique de sharing),
- et enfin, nous avons appliqué quelques résultats théoriques récemments établis sur les algorithmes génétiques à propos des probabilités de mutations, ce qui nous a permis d'améliorer, notamment, les temps d'exécution.

Mots-Clés : Algorithmes génétiques, Extraction de primitives, Sharing, Transformée de Hough.

A Genetic Algorithm for the Detection of 2D Geometric Primitives in Images

Evelyne LUTTON, Patrice MARTINEZ
INRIA - Rocquencourt
B.P. 105, 78153 LE CHESNAY Cedex, France
Tel : 33 1 39 63 55 23 - Fax : 33 1 39 63 53 30
email : evelyne.lutton@inria.fr

We investigate the use of genetic algorithms (GAs) in the framework of image primitives extraction (such as segments, circles, ellipses or quadrilaterals). This approach completes the well-known Hough Transform, in the sense that GAs are efficient when the Hough approach becomes too expensive in memory, i.e. when we search for complex primitives having more than 3 or 4 parameters.

Indeed, a GA is a stochastic technique, relatively slow, but which provides with an efficient tool to search in a high dimensional space. The philosophy of the method is very similar to the Hough Transform, which is to search an optimum in a parameter space. However, we will see that the implementation is different.

The idea of using a GA for that purpose is not new, Roth and Levine [29, 28] have proposed a method for 2D and 3D primitives in 1992. For the detection of 2D primitives, we re-implement that method and improve it mainly in three ways :

- by using distance images instead of directly using contour images, which tends to smoothen the function to optimize,
- by using a GA-sharing technique, to detect several image primitives in the same step,
- by applying some recent theoretical results on GAs (about mutation probabilities) to reduce convergence time.

Keywords : Genetic Algorithms, Image Primitive extraction, Sharing, Hough Transform.

1 INTRODUCTION

Geometric Primitives extraction is an important task in image analysis. For example, it is used in camera calibration, tridimensional stereo reconstruction, or pattern recognition. It is important especially in the case of indoor vision, where most of the objects to be analysed are manufactured. The description of such objects with the help of bidimensional or tridimensional geometric primitives is well adapted.

Our aim is to present an alternative to the well-known Hough transform [18], widely used for the primitive extraction problem. The Hough transform is a very efficient method for lines of simple primitives detection (see for example [22, 30, 24]), but reaches its limits when we try to extract complex primitives. The method consists in the searching of maxima in the space of parameters which describe the primitive. For example to extract circles, we have to construct an accumulator of dimension 3, which becomes very expensive in memory.

The Hough Transform constructs explicitly the function to optimize, which is represented by an "accumulator", i.e. a sampling of the parameter space with "cells". This accumulator can be filled in with two equivalent techniques :

- *the 1 to m technique*, where for one point of the image, we draw (or update the cells) a curve in the parameter space, which represents the parameters of all the primitives which the considered image point may belong to,
- *the m to 1 technique*, also called *randomized* or *combinatorial Hough Transform*, where for all possible m-uple of image points (couple of points for the line detection), we draw a point in the parameter space, which represents the unique primitive that can pass trough the considered m-uple of image points.

The effective detection of primitives is thus done by a rough sequential search on the accumulator. To summarize, the Hough Transform is a very quick and precise technique for simple geometrical primitives, but it becomes rapidly untractable to store an accumulator and detect optima on it when the number of parameters to estimate increases.

This is why we have to think about efficient optimization techniques to solve the problem for complex geometric primitives. As we have seen, it can be easily formulated as an optimization problem : optimizing the position and size of a geometric primitive (or equivalently the values of parameters), knowing the edges detected on an image. The function optimized in the Hough Transform is a function of the parameters, that is the total number of image points which coincide with the trace of the primitive defined by these parameters. Another problem is that, when the dimension of the space to search is large, the function to be optimized can be very irregular.

When a function has a certain type of regularity, a number of optimization methods exists, mostly based on gradient or generalized gradient computations (see for instance [5]).

Generalized gradient methods work well when :

- some sort of gradient can be defined and computed at any point of the space of solutions (for instance, directional derivatives),
- the function does not have too many local minima, or the value taken by the function at these minima is significantly greater than the value at the absolute minimum.

For very irregular functions, different methods have to be used for optimization. Most of them are based on stochastic schemes.

One of the most known stochastic algorithms is Simulated Annealing. It is a powerful technique for finding the global minimum of a function when a great number of parameters have to be taken into account. It is based on an analogy with the annealing of solids, where a material is heated to a high temperature, and then very slowly cooled in order to let the system reach its ground energy. The delicate point is not to lower the temperature T too rapidly, thus avoiding local minima. Application to other optimization problems is done by generalizing the states of the physical system to some defined states of the system being optimized, and generalizing the temperature to a control parameter for the optimization process ; most of the time, the Metropolis algorithm is used : at "temperature" T, the jump from a state of energy E to a state of energy E' is made with probability of one if E' is lower than E and with a probability proportional to $e^{(E-E')/T}$ otherwise ([1, 27]).

The main drawback of Simulated Annealing is the computational time : the optimal solution is guaranteed only if the temperature is lowered at a logarithmic rate([11]), implying a huge number of iterations. Most of the time, a linear rate is used to obtain affordable converging times, but, for certain very wild functions, the logarithmic rate has to be used.

In this work, we investigate the use of another recently introduced method for stochastic optimization, namely Genetic Algorithms [16, 29, 28]. In section 2, we recall the main aspects of genetic algorithms. We then present in section 3 the application to the image primitives extraction problem. In section 4, we introduce the Sharing method, which enables an improvement of the efficiency of our method, and sum up our results in section 5, proposing various desirable extensions.

2 GENETIC ALGORITHMS

Genetic Algorithms can be considered as stochastic optimization methods, but it must be pointed out that they also have other fields of applications, as for instance in neural nets, classifier systems, automatic programming, graph theory, etc ... (see [15, 28, 2, 31, 21, 12, 7]). So for, they have not been largely used in vision and image analysis applications.

The benefit of using Genetic Algorithms to optimize irregular functions is that they perform a stochastic search over a large search space, by making a set of solutions (called *population*) evolve together, instead of using a single solution as in the Simulated Annealing scheme.

We use here a sequential implementation of a Genetic Algorithm, but notice also that GA have the great advantage to be easily parallelized.

The specificity of genetic algorithms is that they try to copy simple natural evolution schemes. John Holland [16] is largely recognized as the founder of the field of Genetic Algorithms. He integrated and elaborated two themes : the ability of simple representations (bit strings) to encode complicated structures, and the power of simple transformations to improve such structures. Holland showed that with the proper control structure, rapid improvements of bit strings could be made to "evolve" as population of animals do. An important formal result stressed by Holland was that even in large and complicated search spaces, given certain conditions on the problem domain, genetic algorithms would tend to converge to solutions that are globally optimal or nearly optimal (Schema Theorem, see [12]).

In natural evolution, the problem each species faces is the one of searching for beneficial adaptations to

a complicated and changing environment. The "knowledge" that each species has gained is embodied in the makeup of the chromosomes of its members. The operations that alter this chromosomal makeup are applied when parents reproduce ; among these operations are random mutation, inversion of chromosomal material, and crossover - exchange of chromosomal material between two parents' chromosomes. This feature of natural evolution - the ability of a population of chromosomes to explore its search space in parallel and combine the best findings through crossover - is exploited when genetic algorithms are used. A genetic algorithm to solve a problem is commonly described as having 5 components (see [8]) :

1. *a chromosomal representation of solutions to the problem,*
2. *a way to create an initial population of solutions,*
3. *an evaluation function that plays the role of the environment, rating solutions in term of their "fitness",*
4. *genetic operators that alter the composition of children during reproduction,*
5. *values for the parameters that the genetic algorithm uses (populations size, probabilities of applying genetic operators, etc ...)*

2.1 Structure of a GA

The general structure of a GA is presented in the figure 1. Of course there exists a lot of variations around this scheme. We present here the classical scheme (Simple Genetic Algorithm, see [12]), which is the basis of our program.

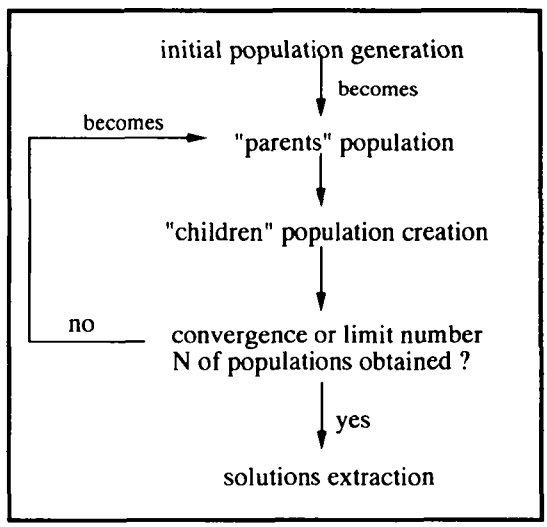
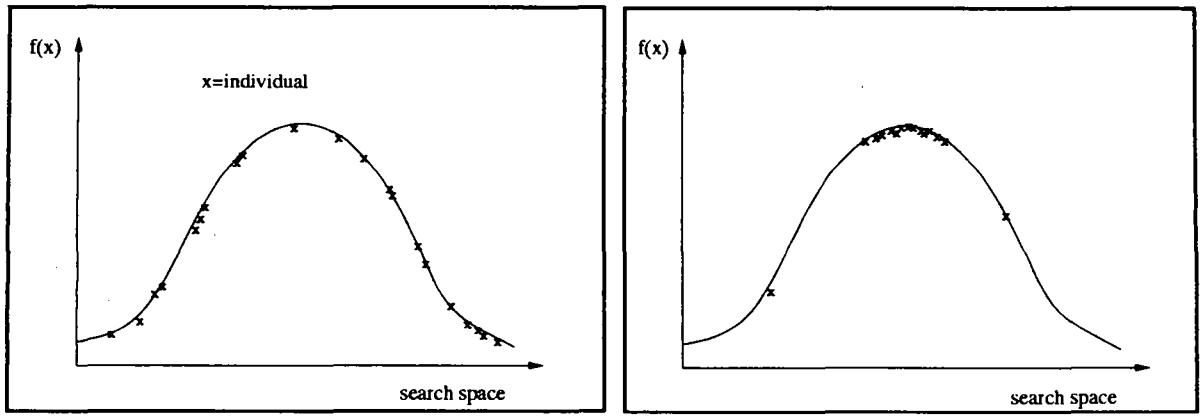


Figure 1: General Organigram of a Genetic Algorithm

The solutions (also called individuals) of the population go through a process of evolution. Some solutions are better than others, in the sense that their fitness function is better. Those that are better are more likely



Random initial population Population after convergence

Figure 2: Evolution of the population

to survive and propagate their genetic material. The convergence of a GA thus leads to a concentration of the population into regions of the search space where the fitness function presents a global optimum, see figure 2.

A GA first needs an initialization procedure of the population. This initial population is usually randomly chosen, albeit sometimes determistically, to be regularly distributed in the search space. Most of the time, we allow the possibility of introducing in the initial population some a priori knowledge, as initial solutions.

The creation of the "children" population is done in three steps : *selection* of two parents according to their fitness, *crossover* of their genetic material to create two offsprings, then *mutation* of the offsprings (see figure 3). The crossover and mutation operators are randomly applied. Crossover is applied with a high probability (namely 0.7 to 0.9), when the outcome of the random draw is negative, the offsprings are just a copy of the parents (we talk about reproduction). Mutation is applied with a very low probability (reversely proportional to the chromosome length) so that few chromosomes are altered.

Selection and Crossover behave as "concentration" operators, they favor the concentration of solutions having good fitness. The risk is to have a premature convergence toward a local minimum. On the contrary, Mutation acts as a "dispersion" operator, that maintains the diversity of the genetic materials. The simultaneous action of these three operators allows to converge into a global minimum. The effects of the genetic operators have been pointed out by Holland through his Schema Theory (see [16, 12]). Recently Markov Chain approaches enable to demonstrate the effects and efficiency of these operators [17, 9, 26]. Moreover, Davis in 1991 [9] has proposed a Simulated Annealing - like convergence proof for the simple Genetic Algorithm, and derived a decreasing formula (very slow) for the mutation probability that guarantees the convergence towards a global optimum.

The problem of stopping the evolution is a difficult one ; of course we can test whether the population is "concentrated", but sometimes this state is difficult to obtain. The most commonly adopted solution is to impose a maximal number of generations.

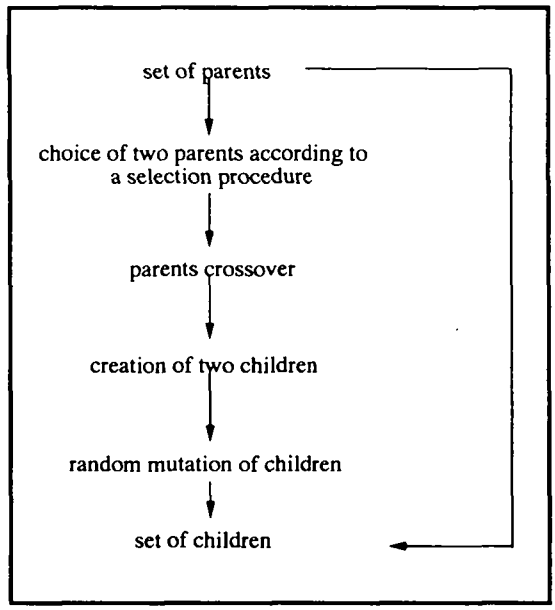


Figure 3: Children creation procedure

2.2 The Chromosomal Representation

In a GA approach the correspondence between the solutions (phenotypes) and the chromosomes (genotypes) is straightforward ; it is most generally a bijection. It is well-known that in nature this equivalence between phenotypes and genotypes is not so simple. There exist, for example, genes not expressed in the phenotype.

Holland [16] proves that the optimal representation for a GA is a code with minimal alphabet, thus the binary codes are well adapted. Later, Goldberg [12] explains that the most efficient code is a compromise between the size of the alphabet and the complexity of the code. Complexity for a code in that case is the fact that small changes on the code can induce large modifications on the solution they represent. Another important constraint is the computational complexity of the encoding/decoding process, because these operations are called many times all during the evolution of a GA. The theoretical knowledge about coding influence on the efficiency of GA is small. Practically, a code is empirically validated.

For the purely optimization applications of GA, the chromosome is simply a concatenation of the binary codes of all the components of the vector of the space to be searched. This space must therefore be bounded. If there are real values, they are sampled with a certain precision, for the case of natural numbers the binary code is straightforward. Notice also that the choice of sampling rates for real values can be problematic [23], and must be handled with care.

2.3 Selection

The selection of a couple of parents is done accordingly to their fitness (i.e. their adaptation to the environment). The function to optimize, which is used as "fitness" function, must be positive ; this is the only constraint on it. It does not need to be derivable nor continuous.

Figure 4 shows a classical selection scheme, where we consider a uniform random shot on a disk, where solutions share sectors proportional to their fitness. The best ones are more often selected than the others.

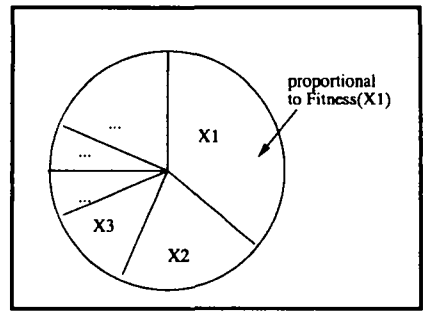


Figure 4: Parents Selection : Roulette Wheel

The problem of this type of selection is due to the fact that it can favor some "super individual" which are almost always selected, thus favoring some premature convergence of the population, by some sort of "lost of genetic diversity".

To have a uniform and reasonable "selection pressure", a current solution is to scale linearly the fitness function so that its maximum value is C times the mean fitness of the current population. C is named the *selection pressure* (for precisions, see [12], C is generally between 1.2 and 2). The probability is thus computed by :

$$P(x) = \frac{f'(x)}{\sum_{x \in Population} f'(x)} \quad \text{and} \quad f'(x) = a * fitness(x) + b$$

$$a = \frac{(C-1) * Favg}{Fmax - Favg} \quad \text{and} \quad b = \frac{Favg * (Fmax - C * Favg)}{Fmax - Favg}$$

$Fmax$ and $Favg$ are the maximum and mean fitness on the current population (see figure 5). Notice that the mean fitness in the scaled population is the same as in the original population, and also that the scaling factors are computed at each generation.

2.4 Genetic Operators

As we have seen, the genetic operators are the basis of the evolution scheme. They make the population to evolve and converge, while allowing a large exploration of the search space.

- **Crossover :**

  The crossover operator makes the population to converge around solutions with high fitnesses. Thus the closer the crossover probability is to 1, the faster is the convergence. The choice of this probability is a compromise, and depends on the form of the evaluation function. It is chosen empirically, most of the time between 0.5 and 0.9.

  There exist two important types of crossover : *the one site crossover*, where a crossover site is randomly chosen, and around which the code chains are exchanged, see figure 6 ; or the *uniform crossover*, where

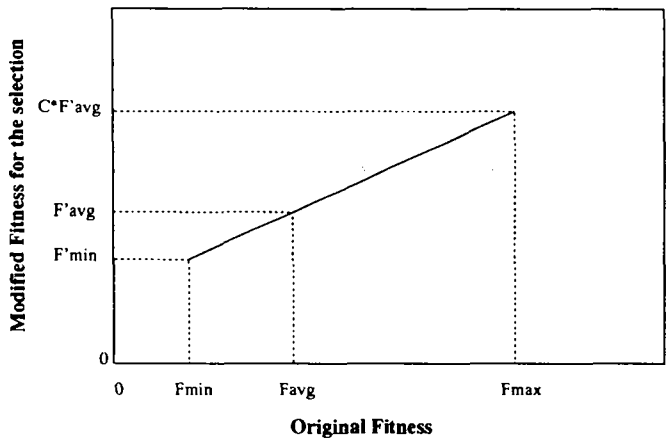


Figure 5: Fitness Scaling for the Selection

each gene of the first offspring is randomly chosen between the parents, the genes of the second offspring is complementary with respect to the random choice, see figure 7. There exists a lot of variations around these types, as the multi-site crossover for example. We have restricted our tests to these two particular ones.

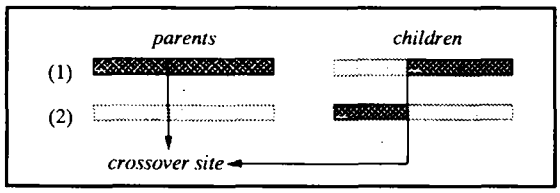


Figure 6: One site Crossover

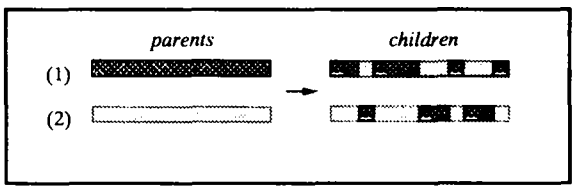


Figure 7: Uniform Crossover

- **Mutation :**

The mutation operator chooses randomly a mutation site, and inverts the corresponding bit (or gene), see figure 8.

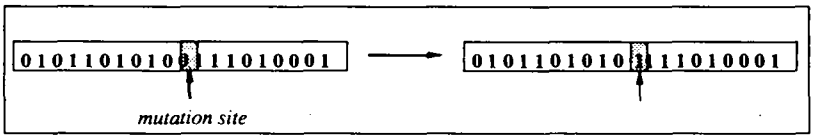


Figure 8: Mutation

The effect of this operator is to "trouble" the convergence tendency in order to let the possibility to visit other regions of the search space. One can tell that it limits the "genetic drift" due to the elitist selection procedure (see [14] for a more theoretical explanation).

The mutation probability must be very low and is, in most of the applications, fixed all along the evolution of the GA. Recently, Davis [9] proposed a decreasing mutation probability with respect to the generation evolution, which assures the theoretical convergence toward the global minimum of a simple GA with finite population size and fixed crossover probability. He proved that the mutation probability $p_m(k)$ must vary at each generation k with respect to a monotonic lower bound :

$$p_m(k) = \frac{1}{2} * k^{-\frac{1}{M*L}}$$

M is the population size and L is the length of the chromosomes.

Of course such a decreasing rate is very slow (see figure 9), and needs an infinite number of generations to make the GA to converge. When we practically implement a Simulated Annealing algorithm, we use a faster decreasing rate of the temperature than the theoretical one. Here we propose a decreasing rate which is given by the formula :

$$p_m(k) = p_m(0) * \exp(-\frac{k}{\alpha})$$

$p_m(0)$ is the initial mutation probability, α is computed to yield a very low mutation probability (namely 10^{-4}) at the end of the evolution :

$$\alpha = \frac{\text{Max Nb of Generations}}{\ln \frac{p_m(0)}{10^{-4}}}$$

The continuous curve of figure 9 is drawn for a Maximal number of Generation of 100 and an initial mutation probability of 0.25. The theoretical curve (dotted) corresponding to a length of chromosome of 32 bits is drawn for comparison with 1000 generations.

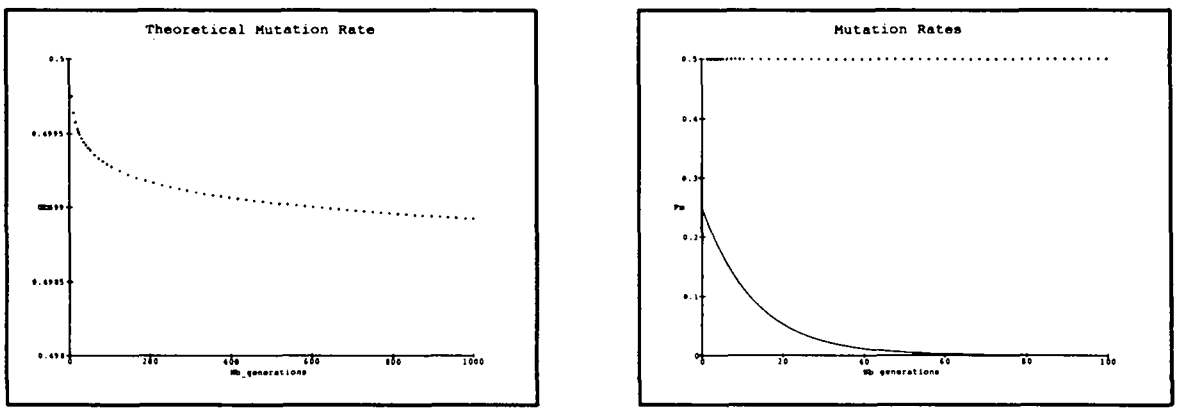


Figure 9: Left, the theoretical curve of Davis (dotted) on 1000 generations, Right, comparison with the one we adopt (continuous) on 100 generations

The decreasing rate that we propose permit to have a large exploration of the search space at the beginning, and then a rapid convergence at the end of the evolution of the GA. This rather rough decreasing rate is

of course dependent on the form of the function to optimize. If this function is too irregular, some slower decreasing rate is necessary, just as in the case of Simulated Annealing.

2.5 Creation of the new population

To create a new population several techniques are frequently used :

- take all the children created by the application of the genetic operators, and let the old generation "die",
- keep some particular solutions of the "old" population (as the best one for example) and complete the "new" generation by applying the genetic operators,
- create a "children" generation of size N (same as the "old" one), and create the "new" generation by keeping the N best solutions of both generations, "old" and "children".

Our tests prove that the last technique is more efficient in our application. It accelerates the convergence speed by allowing to keep good solutions from one generation to the next one. We have also imposed an unicity condition to the population, i.e. no identical solutions can survive together in the same generation. This is done to avoid creations of super-individuals (artificially replicated in the population, due to their good fitness), which favor also the premature convergence.

2.6 Analysis of the final population : solutions extraction

Another interesting characteristic of the convergence of GA is that they "visit" several good local optima, before converging to the global one. A finer analysis than simply finding the solution with best fitness of the final population, can give some interesting information when the GA is stopped before complete convergence.

This can be important, especially in our case, where we search for several local - or global - optima. Thus, we have developed a simple clustering technique, which permits to locate several local optima present in the final population. We will also see (in section 4) that we can favor such convergence comportment using sharing techniques.

The extraction procedure is very simple :

1. we search for the solution having the best fitness, this optimum is the "center" of a cluster,
2. we extract from the population all the solutions which are in the neighborhood of this optimum. This notion of neighborhood is defined with a maximal distance (we will see that this distance is easily defined with the sharing technique)
3. start again in 1, until there is no more solutions in the population, or the fitness of the local maximum is smaller than a fixed threshold.

2.7 Summary

We can summarize the characteristic of a GA in the following points :

- GAs are stochastic optimization techniques, working simultaneously on a set of solutions.

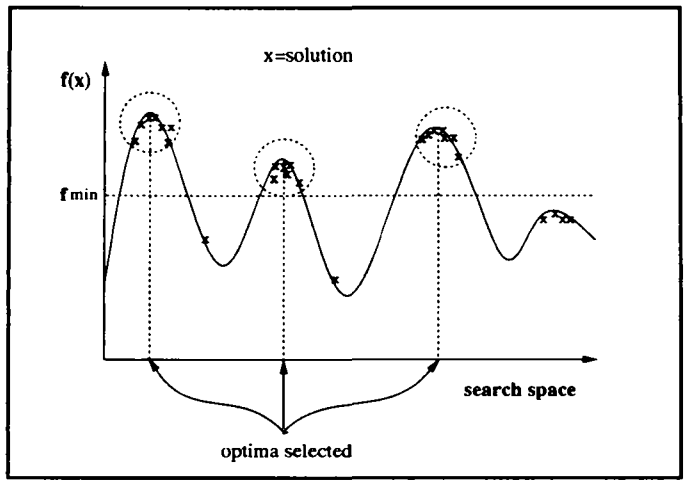


Figure 10: Clustering of the population around several optima

- Only one hypothesis is necessary for the function to be optimized : it must return bounded from below (it is possible to offset the function so that it furnishes a positive or null value to the selection process), no derivability nor continuity hypotheses are needed.
- The convergence of a GA is a concentration of the population near the global optimum.
- The speed of convergence and the quality of the final solution depend on the chromosomal coding, on the form of the function to be optimized, and on the parameters of the GA. The parameters tuning is mostly empirical (few theoretical results exist now).
- In the serial version, the computing time is high, and for applications as ours, most of the time is spent in the computation of the fitness function.

3 THE PRIMITIVE EXTRACTION PROBLEM

We detail here the characteristics of the GA we have implemented for the 2D geometrical primitives extraction application.

3.1 Primitives Coding

In the approach of Roth and Levine [28], the chromosome representing a primitive is the coding of the points needed to define that primitive. These points are contour points of the primitive. The representation of a primitive with a minimal set of points makes the extraction process less noise-sensitive and the chromosomal coding trivial : a chromosome is the concatenation of the codes of its minimal set of points. But that representation has a main drawback, the redundancy : the same primitive can be represented by several differents chromosomes, one for each possible permutation of the points of the minimal set.

In our application, we propose a different coding, which insures the unicity of the representation :

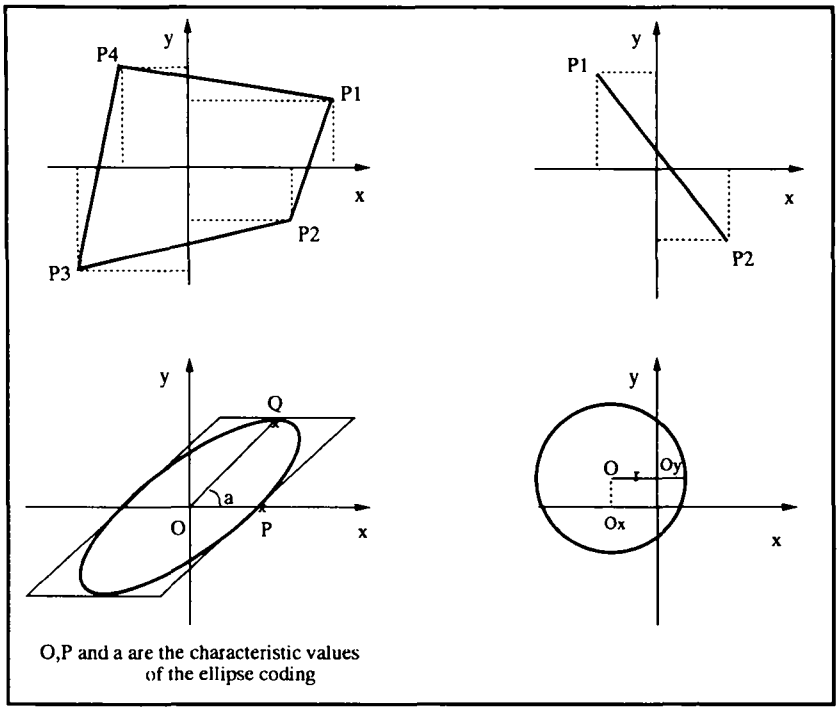


Figure 11: Graphical coding of a segment, a circle, a quadrilateral, and of an ellipse

- **Segment** : 2 points of the image $I = [0..x_{max}, 0..y_{max}]$, with integer coordinates, representing the vertices of the segment,
- **Circle** : 1 point of I for the center of the circle, and a positive integer of $[0..\max(x_{max}, y_{max})]$ for its radius,
- **Ellipse** : 2 point of I, the center O and the point P, a positive real a, between 0 and $\frac{\pi}{2}$, representing the rotation angle of the ellipse. The characteristics of this coding are classical for computer graphics and are detailed in [19], see figure 11,
- **Rectangle** : 2 points of I, coordinates of the top-left and bottom-right vertices. This rectangle is parallel to the axes of the image. For different orientations, we add a positive real, between 0 and $\frac{\pi}{2}$, for the rotation angle.
- **Quadrilateral** : 4 points of I, for the 4 vertices.

3.2 Computation of the fitness function

For the computation of fitness function we prefer to use distance images, instead of simple contour images. Indeed, if we use contour images, the fitness function is a measure of the contour points in coincidence with the trace of the primitive. See figure 12 a), for the example of segment fitness computation. To tolerate small

errors, it is often necessary to make a computation on a strip centered on the primitive, which increases the computational time of an evaluation of the fitness function, see figure 12.

Moreover, the form of the fitness function in that case is very irregular, and a primitive near a real contour has no more information than a primitive which is far away from it. The convergence of a GA in that case can be very slow, especially when the contours are sparse in the image.

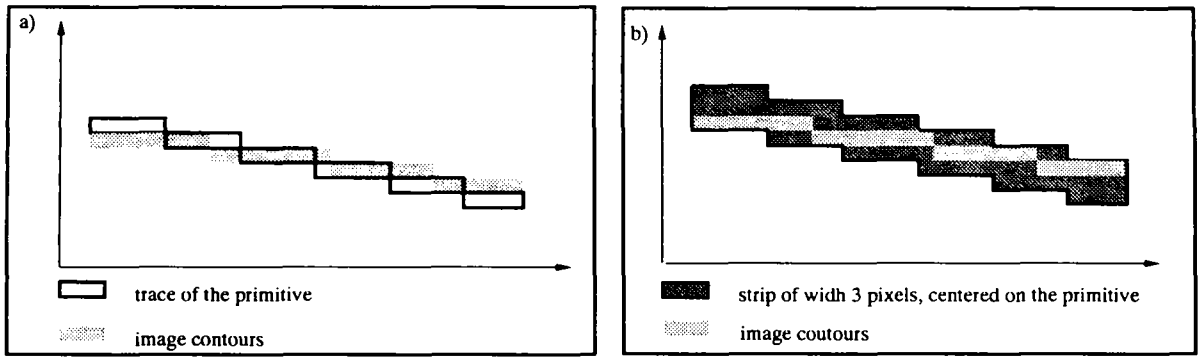


Figure 12: Fitness computation on contour images

We use a well-known tool of mathematical morphology, which furnish distance images, i.e. grey-level images computed from contour images, where each pixels gives the value of its distance to the nearest contour point. These images are easily created by application of two masks on a contour image, see [4]. The distances computed are parameterized by $d1$ and $d2$, see figure 13, which represent the two elementar distances on vertical/horizontal and diagonal directions. We use Chamfer distance ($d1 = 3$ and $d2 = 4$), or more "abrupt" distances ($d1 = 10$ and $d2 = 14$), in our application, see figures 14 and 15. The tuning of $d1$ and $d2$ depends on the frequency of the contour points on the image, and we can think of an "adaptative" tuning of these parameters (we have not implemented it yet).

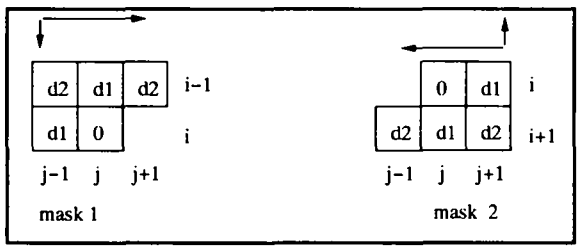
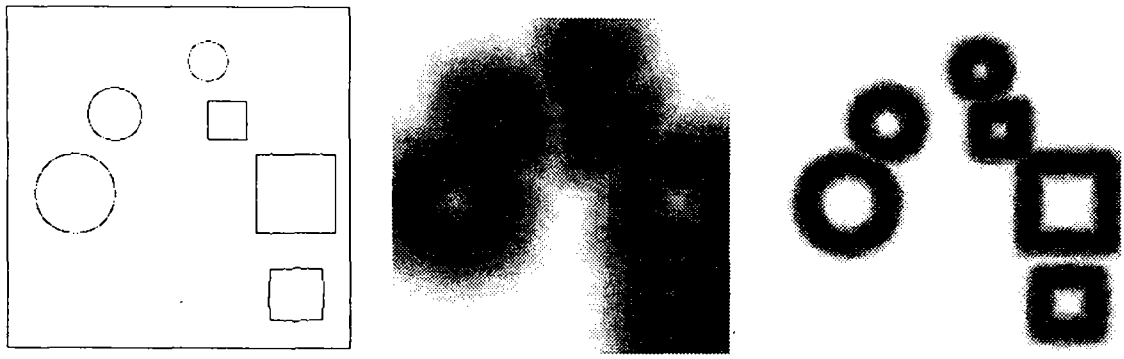


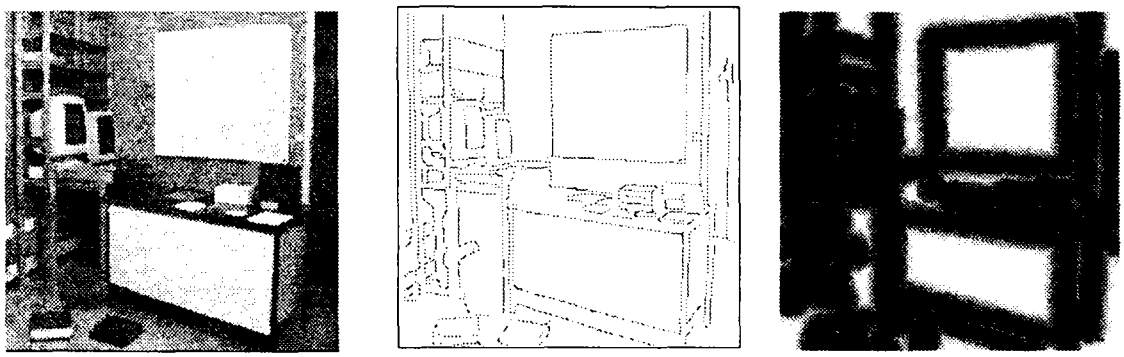
Figure 13: Distance Masks

The benefit of using such distances images is double : first, the fitness function is more rapid to compute (using the trace of the primitive is largely sufficient), and secondly, the tolerance to small errors is improved. The fitness function takes into account the mean intensity of the pixels of the distances image in coincidence with the trace of the primitive (to position the primitive), plus a counting term of effective contour pixels on the trace (to favor bigger primitives).



Contours image | Chamfer distance (3,4) | distance (10,14)

Figure 14: Example of distance images on a synthetic image



Original Image | Contours image | distance (10,14)

Figure 15: Example of distance images on a real image

4 THE SHARING TECHNIQUE

The primitive extraction method described before permits to detect only one primitive at each run of the GA. To detect all the primitives of the image it is necessary to iterate the process, by updating the contour image (just by "removing" the contours on the image that correspond to the detected primitive), regenerate the contour image, and re-run the GA on that new environment. We stop the process when there are no more contours in the image, or when the best detected primitive by the GA has a fitness under a fixed threshold.

The interest of detecting several primitives in the same GA-run is evident. For that purpose, we propose to use a sharing technique, followed by the simple clustering we have described before.

4.1 Historical aspects of the sharing scheme

As we have seen, in the case of multi modal functions, the simple GA is not fully efficient : if there exists several strictly equivalent global optima, the GA population converges randomly to one of them. A solution to that problem is copied from the natural phenomena of "niching" of populations. Individuals of a same subpopulation

have to share the local resources. Due to overcrowding, the local resources decrease, and individuals tend to search other places. In GAs several solutions have been proposed, based on explicit or implicit creation of niches. More precisely, we can divide these approaches in two classes. The first one represents techniques to maintain the diversity in the population along the GA evolution, thus in a certain measure it favors the creation of separate subpopulations. The second one uses a modification of the fitness function to simulate the sharing of local resources in the population.

- **Diversity conservation :** Caviccio's approach in 1971 [12, 13] was the first attempt to induce niche-like and species-like behavior in GA. Specifically, he introduces the mechanism of *preselection* : a child replaces only one of its parents, the one which has the lowest fitness, and only if the child's fitness is better. Chromosomes (or strings) tend to replace similar ones, and thus tend to create sub-species.

  De Jong in 1975 [12, 13, 20] proposed a generalization of this technique with the *crowding* scheme. A child replaces one of its neighbor, the one having the lowest fitness. Strings replace existing strings accordingly to their similarity. It tends to maintain diversity within the population, and reserve room for one ore more species.

  Mauldin in 1984 [12, 13, 25] proposed a sort of unicity operator using a Hamming distance. He defines a uniqueness operator, which arbitrarily returns diversity to a population whenever it is judged to be lacking. A child must be different of every population member at a minimum of Ku genes. If it is not sufficiently different, it is mutated until it satisfies the constraint. Ku decreases with time (similar to the cooling of simulated annealing).

  It is interesting to note that uniqueness combined with De Jong's crowding scheme worked better than either operator by itself [13, 25]. But we will see that with help of *sharing functions* it is possible to maintain an appropriate "more intelligent" diversity.

- **Sharing :** Goldberg and Richardson in 1987 [13, 12] proposed the *sharing* scheme, where individuals share effectively the local resources. This is a way of imposing niche and speciation on strings, based on some measure of their distance to each other. This is done with the help of a so-called sharing function. Schematically, the fitness of an individual is lowered in function of the number of its neighbors.

  These methods have been carefully studied these last five years, and the ability of the sharing technique has been theoretically demonstrated to find multiple, good solutions, for example using the finite Markov Chain Analysis as Horn in 1993 [17]. A niched GA tell us more about the fitness landscape than what the best solution is : each niche, representing a "good solution" has a subpopulation proportional to its fitness.

Notice that there exists also techniques that explicitly creates subpopulations, as Cohoon in 1991 [6], on separate processors, and exchanges individuals between these subpopulations at fixed times, creating a sort of "catastrophe" in the stabilized subpopulation (Punctuated Equilibria).

We have preferred to use the sharing technique defined by Goldberg and Richardson, because the preceding niching technique does not ensure that the separate subpopulations will evolve on separate optima. In a certain manner, the sharing technique ensures that several optima will be "inhabited" if there exists.

4.2 The sharing function

The sharing function is a way of determining the degradation of an individual's fitness due to a neighbor at some distance. Of course, we have to define a distance on our search space. It can be computed on chromosomes (genotypic distance), as Hamming distances between strings, or in the search space itself (phenotypic distance).

In our application, we have preferred to use the genotypic distance between primitives. Indeed genotypic distances, when we can use them have been demonstrated as being more powerful [13].

The neighborhood notion is a fuzzy one, we define a fuzzy neighborhood from a membership function $sh()$, which is a function of the chosen distance d. The membership function that we use, proposed by Goldberg and Richardson [13], depends on two constants : σ_{share} which commands the extent of the neighborhood, and α for its "shape", see figure 16

$$Sh(d) = \begin{cases} 1 - (\frac{d}{\sigma_{share}})^{\alpha} & \text{if } d < \sigma_{share} \\ 0 & \text{else} \end{cases}$$

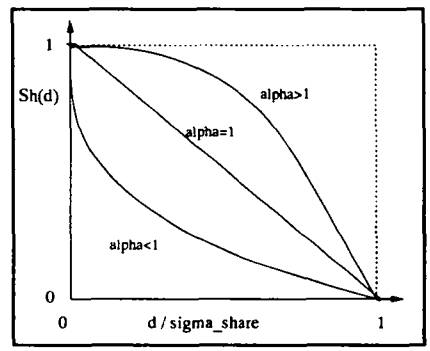


Figure 16: Neighborhood form according to α

Sharing is implemented by changing the fitness function to a shared fitness, which is the fitness divided by its niche count m_i :

$$\text{NewFitness}(i) = \frac{\text{Oldfitness}(i)}{m_i}$$

$$m_i = \sum_{k=1}^{N} Sh(d_{ik})$$

The effect of sharing is to separate the population in subpopulations of sizes proportional to the height of the optima. Goldberg and Richardson [13] have explained that a sharing is stabilized when $\frac{f_h}{m_h} = \frac{f_k}{m_k}$ ($\forall h, k$ with h and k different local optima), see figure 17.

The results of the sharing technique depend mainly on the tuning of the parameter σ_{share}, which is a measure of the separation we accept between two detected peaks of the function to optimize. Notice that methods to estimate the σ_{share} parameter have been proposed (as in [10]).

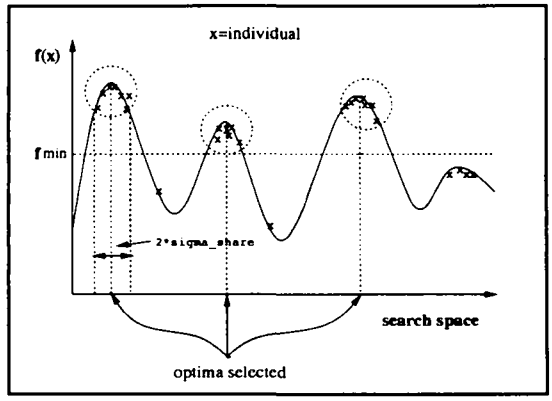


Figure 17: Repartition of the population after a sharing

4.3 Sharing with mating restriction

If we examine the evolution of a shared GA, on a simple multi modal function, we notice that there are some individuals that drift between two peaks. This is due to the fact that the crossover of individual belonging to different niches rarely results in an individual near these optima. Booker [3] has proposed to limit the crossover of different "species". He modified the selection technique in the following way :

- choice of the first parent,
- scanning of the population to find the subpopulation of individuals that have a distance with it less that σ_{cross} (σ_{cross} is currently take equal to σ_{share}),
- if the size of this subpopulation is bigger that 1, we apply the classical elitist selection in that subpopulation, otherwise, the other parent is chosen in the whole population.

Goldberg and Richardson [13] have proven on simple multi modal functions that the mating restriction scheme improves the efficiency of the sharing.

4.4 A modified sharing technique

We propose another scheme for the sharing, based on the relative importance of the individuals in a neighborhood. An individual with a low fitness in a niche will have a few influence on crossovers in that particular sub-population, and will surely disappear to the benefit of better individuals. Goldberg and Richardson just take into account the number of neighbors to share the fitness. We prefer to take also into account the "force" of the neighbors. The new sharing scheme is thus :

$$\text{NewFitness(i)} = \frac{\text{OldFitness(i)}}{\mu_i}$$

$$\mu_i = \sum_{k=1}^{N} (\text{OldFitness}(k) * Sh(d_{ik}))$$

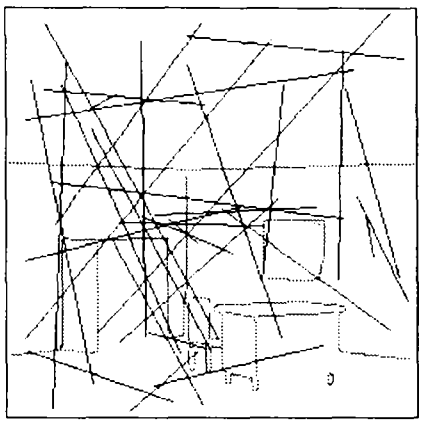


Figure 18: Initial random population (segments)

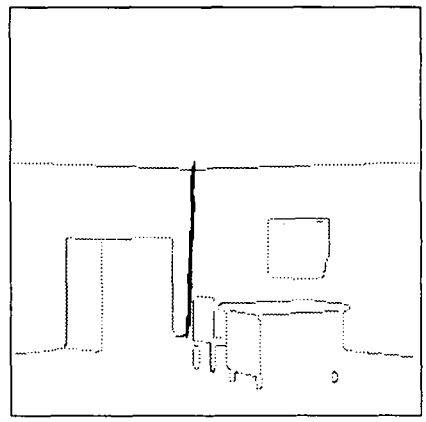


Figure 19: Final classical convergence of the population (after 80 generations)

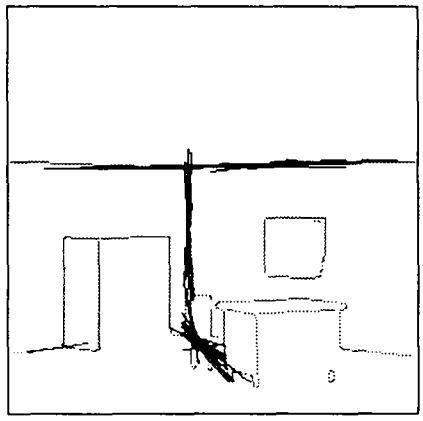


Figure 20: Convergence with sharing

μ_i represents a fuzzy mean fitness in the neighborhood of the individual i. Thus $\frac{fitness(i)}{\mu_i}$ is a measure of the relative importance of the individual with respect to his neighborhood. Following [13], we can tell also that the evolution process is stabilized when :

$$\frac{fitness(pic_h)}{\mu_{pic_h}} = \frac{fitness(pic_k)}{\mu_{pic_k}}$$

It tends to equilibrate the subpopulations in the peaks, independently of their height, since they are bigger than a certain threshold. This particular fact permits to "inhabit" more peaks with the same population size, than with the classical sharing technique, where the best peaks attract much more individuals, and thus reduces the number of individuals which can occupy other peaks.

This comportment is very interesting in our case because the function we have to optimize is strongly multi modal, and we have noticed an important improvement in the performances in using our sharing scheme, in comparison with the classical sharing. For example, for a population of 100 individuals (see figures 18, 19, and

20), we can detect 4 to 7 optima in the same run. Of course the shared GA takes more CPU time to converge but we noticed a global improvement of the computational time in comparison with the simple GA.

5 RESULTS

We present here results on synthetic and real images, for three primitives :

- **segments** : on figures 23 and 31, a GA run (wich furnish 4 to 12 segments at the same time) takes 10 to 15 seconds on a Sparc II station,
- **circles** : on figure 24, it takes 70 to 80 seconds for circles,
- **ellipses** : on figures 32 and 33,
- **rectangles** : on figures 25 and 28.

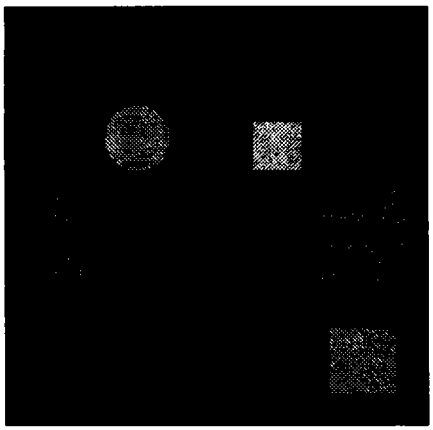


Figure 21: Synthetic image

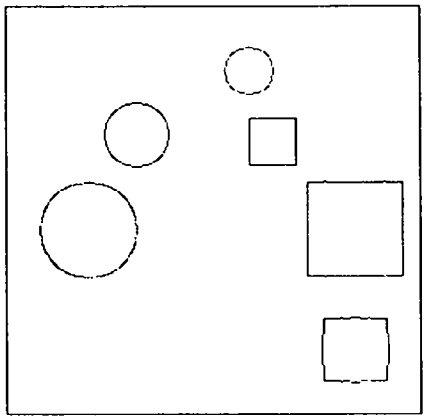


Figure 22: Contours

6 CONCLUSION

The method we have presented, is complementary to the Hough Transform, in the sense that for simple primitives (less than 4 parameters), GA is not efficient, Hough Transform must be used. For the segments detection task, the GA application we have presented (in a parallel implementation), could be concurrent to the Hough Transform. GA is much more interesting for complex primitives (circles, ellipses, quadrilaterals, etc ...). The application to detecting another type of primitive is easily done by updating the fitness function, and the distance function.

We can also think about some applications of the Hough Transform, called Generalized Hough Transform, where we have a reference form (non parameterized) and search the presence of that form in an image. This is done through the construction of an accumulator of possible translations and rotations of the form. Once again the Hough Transform is limited, and the search is mostly done on a few parameters (translation *or* rotation *or*

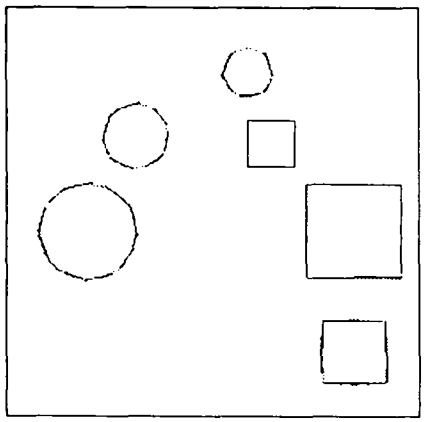


Figure 23: Segments detected (black) and contours (grey)

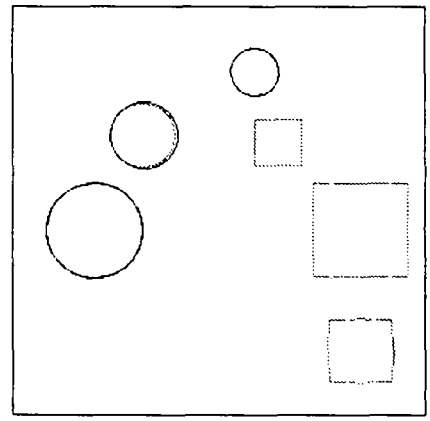


Figure 24: Circles detected (black) and contours (grey)

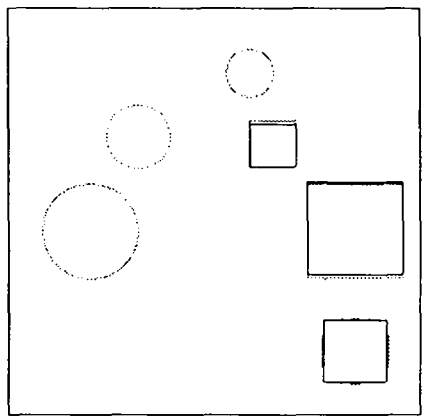


Figure 25: Rectangles detected (black) and contours (grey)

dilatation). The GA approach can also furnish a tool to do such a search with displacement, and deformation parameters together.

The particular formulation of GA approach permits to easily use tools as distance images, which smoothen the function to optimize, and decreases the convergence time. We have also exploited the sharing scheme to improve the efficiency of the search on multi modal fitness functions.

The main problem of such an approach remains the parameters tuning, because it severely influences the convergence speed, and the quality of results. Except for the mutation probability where we could use some theoretical results, this tuning is now experimentally done, and varies for each type of primitives.

To conclude, we can tell that theoretical researches on GAs are directed towards the problem of judicious choice of parameters, see for example the recent results we use for the mutation probability variation ([9] in 1991). Another point to mention is that GAs can be very easily parallelized, and some authors claim that it permits to divide the computational time almost by the number of processors used. We intend now to program a parallel version of the algorithm.

Figure 26: Synthetic image

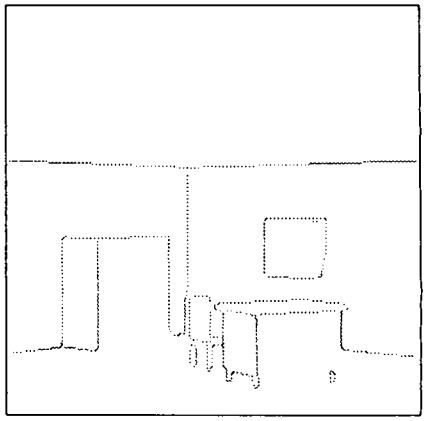


Figure 27: Contours

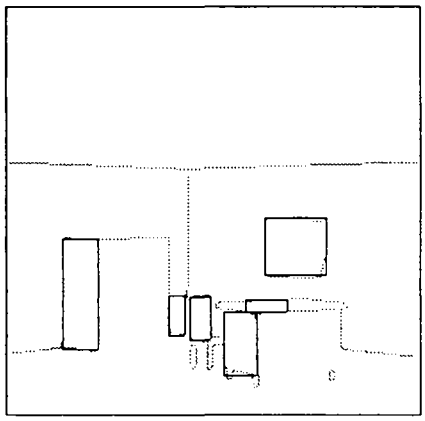


Figure 28: Rectangles detected (black) and contours (grey)

We hope to have pointed out in this paper the interest of considering GA approaches for complex optimization problems involved in image processing and robot vision tasks. We do not claim that GA can replace some well-known techniques, but we think that they can be considered as a complementary approach for some problems which are untractable with classical techniques.

References

[1] E. Aarts and P. Van Laarhoven. Simulated annealing : a pedestrian review of the theory and some applications. AI Series F30, NATO.

[2] J. Albert, F. Ferri, J. Domingo, and M. Vincens. An approach to natural scene segmentation by means of genetic algoritms with fuzzy data. In *Pattern Recognition and Image Analysis*, pages 97–113, 1992. Selected papers of the 4th Spanish Symposium (Sept 90), Perez de la Blanca Ed.

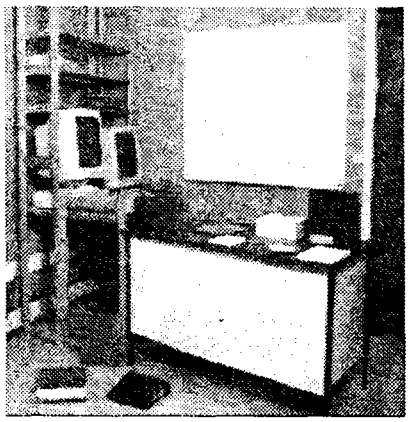


Figure 29: Real image

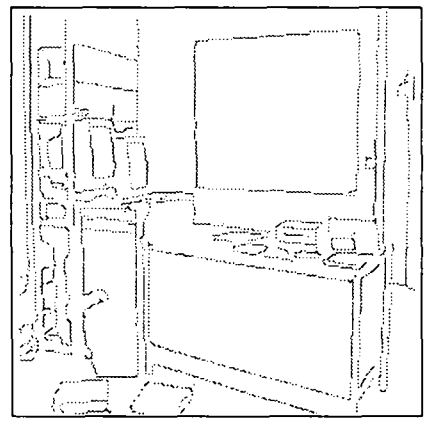


Figure 30: Contours

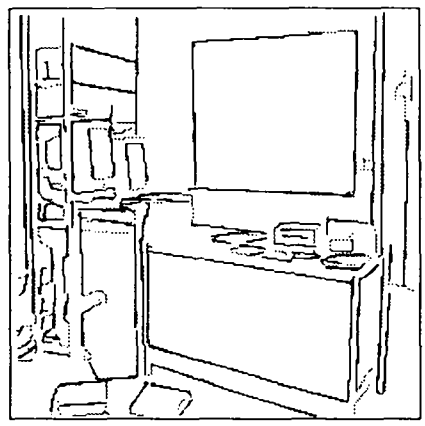


Figure 31: Segments detected (black) and contours (grey)

[3] L. B. Booker. *Intelligent behavior as an adaptatition to the task environment.* PhD thesis, University of Michigan, Logic of Computers Group, 1982.

[4] Gunilla Borgefors. Distance transformations in arbitrary dimensions. *Computer Vision, Graphics, and Image Processing*, 27:321–345, 1984.

[5] A. Bihian C. Lemarechal, J.J. Strodiot. *On a bundle algorithm for nonsmooth optimization*, pages 245–282. Academic Press, 1881. Non-Linear Programming 4, Mangasarian, Meyer, Robinson, Editeurs.

[6] J. P. Cohoon, S. U. Hegde, W. N. Martin, and D. S. Richards. Distributed genetic algorithms for the floorplan design problem. *IEEE Transactions on Computer-Aided Design*, 10(4):483–492, April 1991.

[7] Y. Davidor. *Genetic Algorithms and Robotics. A Heuristic Strategy for Optimization.* World Scientific, 1991.

[8] L. Davis. *Genetic Algorithms and Simulated Annealing.* Pittman, London, 1987.

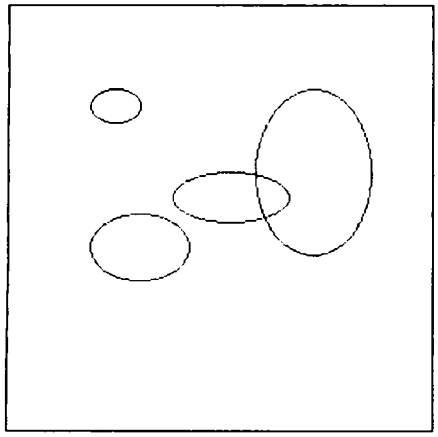


Figure 32: Synthetic contours

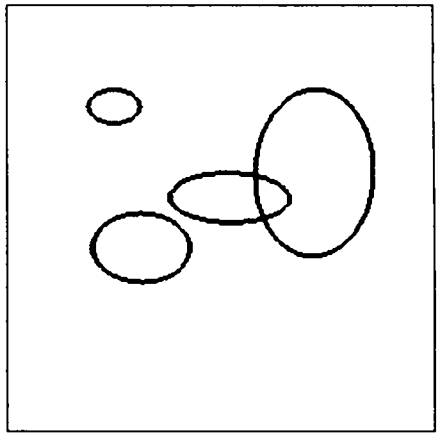


Figure 33: Ellipses detected

[9] T. E. Davis and J. C. Principe. A simulated annealing like convergence theory for the simple genetic algorithm. In *Proceedings of the Fourth International Conference on Genetic Algorithm*, pages 174–182, 1991. 13-16 July.

[10] Kalyanmoy Deb and David E. Goldberg. An investigation of niche and species formation in genetic function optimization. In *Proceedings of the third Conference on Genetic Algorithms*, pages 42–50, 1989.

[11] S. Geman and D. Geman. Stochastic relaxation, gibbs distributions, and the bayesian restoration of images. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 6(6):712–741, November 1984.

[12] D. A. Goldberg. *Genetic Algorithms in Search, Optimization, and Machine Learning.* Addison-Wesley, January 1989.

[13] David E. Goldberg and J. Richardson. Genetic algorithms with sharing for multimodal function optimization. In J. J. Grefenstette, editor, *Genetic Algorithms and their Applications*, pages 41–49, Hillsdale, New Jersey, 1987. Lawrence Erlbaum Associates.

[14] David E. Goldberg and Philip Segrest. Finite markov chain analysis of genetic algorithms. In *Proceedings of Second International Conference on Genetic Algorithms and their Applications*, pages 1–8, 1987.

[15] A. Hill and C. J. Taylor. Model-based image interpretation using genetic algorithms. *Image and Vision Computing*, 10(5):295–301, June 1992.

[16] J. H. Holland. *Adaptation in Natural and Artificial System.* Ann Arbor, University of Michigan Press, 1975.

[17] J. Horn. Finite markov chain analysis of genetic algorithms with niching. IlliGAL Report 93002, University of Illinois at Urbana Champaign, February 1993.

[18] P. V. C. Hough. A new method and means for recognizing complex pattern, 1962.

[19] A. Van Dam J.D. Foley. Computer graphics - principles and practise.

[20] K. A De Jong. *An Analysis of the behavior of a class of genetic adaptive systems.* PhD thesis, University of Michigan, 1975.

[21] J. R. Koza. *Genetic Programming.* MIT Press, 1992.

[22] V. F. Leavers. The dynamic generalized hough transform for the concurrent detection of circles and ellipses. In *Progress in Image Analysis and Processing II.* 6th International Conference on IASP, 1991.

[23] E. Lutton and J. Lévy Véhel. Optimization of fractal functions using genetic algorithms. 1993. Fractal 93, Londres.

[24] Evelyne Lutton, Henri Maitre, and Jaime Lopez-Krahe. Determination of vanishing points using hough transform. *PAMI*, 1993. to appear in.

[25] M. L. Mauldin. Maintening diversity in genetic search. In *Proceedings of the National Conference on Artificial Intelligence*, 1984.

[26] A. E. Nix and M. D. Vose. Modeling genetic algorithms with markov chains. *Annals of Mathematics and Artificial Intelligence*, 5:79–88, 1992.

[27] R. Otten and L. Van Ginneken. Annealing : the algorithm. Technical Report RC 10861, March 1984.

[28] G. Roth and M. D. Levine. Geometric primitive extraction using a genetic algorithm. In *IEEE Computer Society Conference on CV and PR*, pages 640–644, 1992.

[29] Gerhard Roth and Martin D. Levine. Extracting geometric primitives. *CVGIP: Image Understanding*, 58(1):1–22, July 1993.

[30] Frank Tong and Ze-Nian Li. On improving the accuracy of line extraction in hough space. *International journal Of Pattern Recognition and Artificial Intelligence*, 6(5):831–847, 1992.

[31] S. Truvé. Using a genetic algorithm to solve constraint satisfation problems generated by an image interpreter. In *Theory and Applications of Image Analysis : 7th Scandinavian Conference on Image Analysis*, pages 378–386, August 1991. Aalborg (DK).

Unité de Recherche INRIA Rocquencourt
Domaine de Voluceau - Rocquencourt - B.P. 105 - 78153 LE CHESNAY Cedex (France)

Unité de Recherche INRIA Lorraine Technopôle de Nancy-Brabois - Campus Scientifique
615, rue du Jardin Botanique - B.P. 101 - 54602 VILLERS LES NANCY Cedex (France)
Unité de Recherche INRIA Rennes IRISA, Campus Universitaire de Beaulieu 35042 RENNES Cedex (France)
Unité de Recherche INRIA Rhône-Alpes 46, avenue Félix Viallet - 38031 GRENOBLE Cedex (France)
Unité de Recherche INRIA Sophia Antipolis 2004, route des Lucioles - B.P. 93 - 06902 SOPHIA ANTIPOLIS Cedex (France)

EDITEUR
INRIA - Domaine de Voluceau - Rocquencourt - B.P. 105 - 78153 LE CHESNAY Cedex (France)

ISSN 0249 - 6399