Les Cahiers du GERAD ISSN: 0711–2440

Improving the Probability of Success of Repeated Genetic Algorithm on Affine Object Location Problem

B.K.S. Cheung, S.Y. Yuen,
C.K. Fong

G–2005–68

September 2005

Improving the Probability of Success of Repeated Genetic Algorithm on Affine Object Location Problem

Bernard K.S. Cheung

GERAD and École Polytechnique de Montréal
C.P. 6079, Succ. Centre-ville
Montréal (Québec) Canada H3C 3A7
bernard.cheung@gerad.ca

Shiu Yin Yuen and Chun Ki Fong

Department of Computer Engineering and Information Technology
City University of Hong Kong
83, Tat Chee Avenue
Kowloon Tong, Hong Kong, PRC
itkelvin@cityu.edu.hk

September 2005

Les Cahiers du GERAD

G–2005–68

Abstract

Affine object location is a difficult problem in computer vision. Genetic algorithm (GA) provides an efficient solution to the problem when there is little noise or other artifacts. Nonetheless, there is a need to design better GA for complicated, noisy images. In this paper, we investigate the performance of improved operators in GA designs to solve the above problem. Three operators are investigated: redundancy checking, adaptive mutation and partial reshuffling. An intuitive discussion of why these operators are helpful is given in terms of the basic working principle of the GA. Experimental results on the probability of success are given using the framework of Repeated Genetic Algorithm (RGA), which is an application of the probabilistic amplification technique. Tests on both synthetic and real images, with random and structured noise are conducted. It is found that whilst all three operators can improve the GA's probability of success for the affine object location problem, the redundancy checking operator is the most effective.

Résumé

La localisation d'un objet affin est un problème difficile en vision automatisée. Un algorithme génétique fournit une solution efficace pour ce problème quand il y a peu de bruit et peu d'autres objets. Néanmoins, le besoin se fait sentir de créer de meilleurs algorithmes génétiques pour les images compliquées et bruitées. Dans cet article, nous étudions la performance d'opérateurs améliorés dans les algorithmes génétiques pour résoudre le problème décrit ci-dessus. Trois opérateurs sont examinés pour détecter les croisements redondants, la mutation adaptative et la régénération partielle. Sur la base du principe général de fonctionnement de l'algorithme génétique, nous discutons intuitivement pourquoi ces opérateurs sont utiles. Des résultats expérimentaux sur la probabilité de succès sont donnés, en utilisant la structure de l'algorithme génétique répété, celui-ci étant une application de la technique de l'amplification probabiliste. Des expériences sont faites sur des images réelles et synthétiques avec des bruits aléatoires ou structurés. Nous démontrons que même si tous les opérateurs peuvent améliorer la probabilité de succès, l'opérateur qui détecte les croisements redondants est le plus efficace.

Acknowledgments: This research was supported by a grant from a grant from the Research Grant Council of the HKSAR, China (CityU 1157/00E) and a grant from NATEQ, Quebec, Canada (2003-ER-833260).

1 Introduction

Genetic Algorithm (GA) has been applied extensively in hard optimization problems in many fields of applications [1]. One of the main problems with GA is the slow convergence to the desired solution (i.e. the global optimum) when the landscape is complicated. To alleviate this, Yuen et al. [12] proposed the Repeated Genetic Algorithm (RGA). It relaxes the requirement that the GA *must* converge to the correct solution in every run. Instead, it is only required that a single run of GA (a SGA) converges with a probability P_{SGA}. Then the number of independent GA runs N required to guarantee at least a user defined minimum probability P_{RGA} that the correct solution has been seen at least once is

$$N = \frac{\ln(1 - P_{RGA})}{\ln(1 - P_{SGA})} \tag{1}$$

This is known as the *probability amplification* technique [15]. Let $P_{SGA} \geq \frac{1}{poly(\gamma)}$ for a problem with dimension γ and $poly(\gamma)$ be a polynomial function of γ. If one chooses $N = kpoly(\gamma)$ for some positive integer k, then the expected sequential run time is still polynomial in γ, whereas the probability of not finding the correct solution in N runs decreases exponentially in k [8]. The curve of P_{SGA} over N is shown in Figure 1. From the engineering viewpoint, the RGA can also be implemented efficiently by N parallel GAs, such that the computation time is equal to that of one GA alone. This is useful in time critical applications.

If the probability of a single run GA is only 5%, it would require 59 repeated runs for a $P_{RGA} = 95\%$ success rate, but if this probability is enhanced to 20%, then the number of repeated runs is only 14, a significant saving of computation time (provided the SGA does not take much longer to run). Thus it would be significant to investigate ways to improve the probability of success of a single GA run without great increase in its time complexity. In this paper, we investigate the effects of three different operators and their combined designs in improving the probability of success. The three operators are redundancy check with Hamming distances during crossover [6], adaptive mutation [3] and partial reshuffling [2]. We apply these operators to the challenging affine matching/object location problem in computer vision under random and structured noise.

Object recognition is a fundamental problem in computer vision. 3D object recognition is difficult due to non-rigid shape changes, viewpoint changes, perspective distortion, presence of other occluding objects and illumination effects. Efficient 3D object retrieval from a large database of objects is also a difficult problem.

Typical strategies for object recognition can be divided into deterministic and stochastic techniques. Deterministic techniques such as geometric hashing, invariant indexing Hough transform [13] and tree search [7] are robust but are computationally or memory intensive, or both. The attraction of stochastic techniques such as the GA is that they can quickly arrive at a good solution, at comparatively little computational and memory costs. However, the solution obtained has no guarantee to be globally optimal.

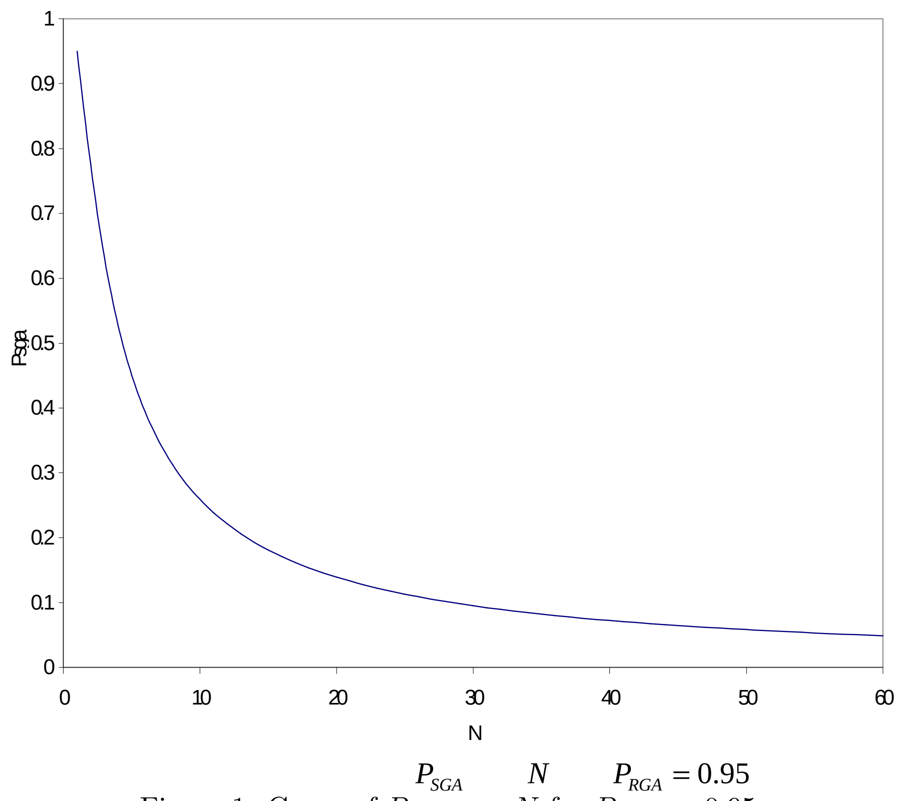


Figure 1: Curve of P_{SGA} vs. N for $P_{RGA} = 0.95$.

The GA has been applied to solve the object recognition problem due to viewpoint changes [9,10,14]. When an object undergoes 3D rigid transformation and weak perspective projection, the 2D image coordinates of a novel view can be expressed as a linear combination of the 2D image coordinates of three reference views. When the 3D transformation is non-rigid but linear, only two reference views suffice. Finally, when the object is planar, then only one reference view is required and the problem is equivalent to the affine object location or matching problem:

In the affine matching problem, we have a known template$T = \{(x_t, y_t)\}$. We wish to find the unknown affine transform $S = (a, b, c, d, e, f)$ such that

$$\begin{pmatrix} x_i \\ y_i \end{pmatrix} = \begin{pmatrix} a & b \\ c & d \end{pmatrix} \begin{pmatrix} x_t \\ y_t \end{pmatrix} + \begin{pmatrix} e \\ f \end{pmatrix} \tag{2}$$

where $I = \{(x_i, y_i)\}$ is a set of edge detected image points. The problem is difficult because of the large search space and the presence of other objects (structured noise) as well as noise outliers (random noise) due to the imperfect edge detection and noise introduced during image formation. The RGA has been applied to this problem [12], but only a simple GA design with standard operators is employed. Two other GAs that have been designed to tackle this problem are [9,10]. The GA in [9] uses an alternative problem coding and that in [10] uses a so called migration principle to improve the performance. However, the fitness function in both papers is area based. This requires a complete contour, which is notoriously difficult to extract in any reasonably complex images. Our GA design also uses the coding in [9] but does not require a complete contour.

This paper is organized as follows: In Section 2, we give an intuitive picture of why the improved operators should be useful. In Section 3, we report the application to the affine object location problem. In Section 4, the experimental results are reported. Section 5 is a conclusion. A preliminary version of this paper appears in [16].

2 Design methodology

The design of GA can not yet be done by theoretical calculations in full rigor. Only worst case bounds [4] or order functions for simple problems are known [5]. In this section, we give an intuitive view behind the design of operators in a GA with a better P_{SGA}.

2.1 Issues affecting the effectiveness of a crossover operation

Let γ be the chromosome length. Let Z_1 and $Z_2 (i = |Z_1|, j = |Z_2|)$ be the subsets of genes in the first and the second chromosomes that are identical to the optimal chromosome respectively. To visualize the effect of a general crossover (e.g. 1-point, 2-point, uniform), one needs only consider the standard case as illustrated in Figure 2, since for fixed i, j and overlap m, any other case is unique up to a permutation of their relative positions.

A crossover exchanges two subsets S_1 and S_2 of the same cardinality each belonging to one of the given chromosomes. This crossover operation can only be effective in producing an offspring closer in resemblance to the optimal chromosome if the exchange involves swapping of genes in $Y_2 = Z_2 \backslash (Z_1 \cap Z_2)$ with genes in the complement of Z_1 in the first chromosome or vice versa. It fails if the subset S_1 of genes selected for crossover in the

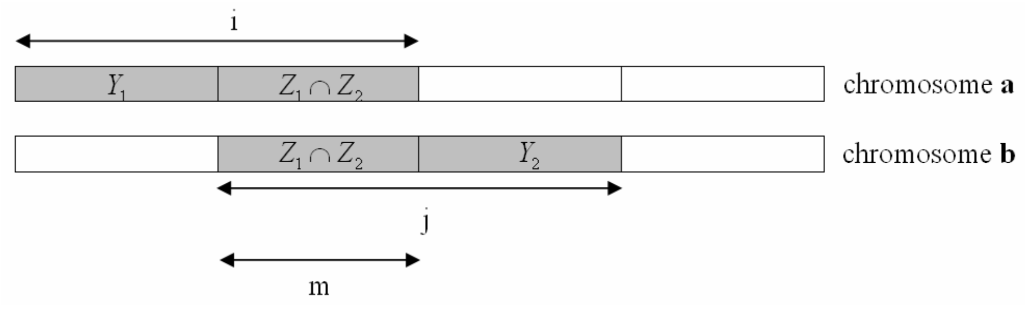


Figure 2: The canonical scenario of a crossover operation.

first string does not contain genes in Z_1 and the subset S_2 in the second string does not contain genes in Z_2. Even if the previous case is not true, it may be entirely redundant if those genes in S_1 and that in S_2 both belong to $Z_1 \cap Z_2$. Furthermore, if a crossover operation gives two children with k attributes and l attributes identical with the optimal chromosome, we must have $i + j = k + l$. Thus, if the number of attributes of one of the offspring (say k) is higher than both of that of its parents, then we must have $k > i > j > l$ (assuming $i > j$ for convenience).

Observe that if the overlap $Z_1 \cap Z_2$ is very small and if both the sizes of Y_1 and Y_2 are close to half of that of the entire chromosome, than any crossover action that swaps most elements of Y_2 and most elements of Y_1 will produce an offspring having most of the attributes of the optimal chromosome. Since the positions selected for crossover is random, the probability of occurrence of this event exists, even though it is rather low.

Following from the previous arguments, we observe:

i) If a pair of distinct chromosomes each has approximately half of the number of attributes of the optimal chromosome, then the chance of obtaining an offspring very close to the optimal chromosome after a single crossover does exist. However, this is true, only if the overlapping of good attributes (i.e. the size of $Z_1 \cap Z_2$) between these two chromosomes is small, otherwise, the probability of obtaining such an offspring will be largely diminished. The worst-case scenario is that Z_1 is contained in Z_2 or vice versa. Its occurrence however, is rare. As a consequence, one should consider *only crossing over pairs of chromosomes that are more distinct* (as measured by their Hamming distance). This procedure helps to avoid most of the redundant operations and at the same time, increases the chance of producing an offspring much closer to the optimum.

ii) It can be seen that at the initial stage of reproduction, most of the chromosomes contain about one half of the attributes of the optimal chromosome. This is true for population with binary strings, where over 80% of the initial population lies within the Hamming distance of 1/4 to 3/4 of the total length of the string to the optimal string. In the case that each of the genes takes up only a narrow range of integer values, the distribution of these attributes may be somewhat shifted to the lower side, but the proportion of chromosomes being close to half of those attributes is still quite large. More precisely, for a chromosome of length γ, where each of its genes assumes v integer values $1, 2, \ldots, v$, the probability distribution of the number of optimal solution attributes in the initial population is given by

$$p(x) = \begin{pmatrix} \gamma \\ x \end{pmatrix} \left(\frac{1}{v}\right)^{x} (1 - \frac{1}{v})^{\gamma - x} \tag{3}$$

Here, the maximum of x should lie between γ/v and $\gamma/2$. When $v = 2$, we have the case of population with binary chromosomes, where x assumes its maximum value equals to $\gamma/2$. It follows from i) that a crossover operation tends to be very effective at the early stages of reproduction.

iii) When the reproduction process reaches its late stages, a large proportion of chromosomes in the population are then very close to the true optimal chromosome. Any crossover action may become very ineffective, as the overlap (i.e. the size of $Z_1 \cap Z_2$) between these two corresponding sets of attributes is large. One could perhaps adjust the crossover rate relative to the mutation rate so as to enhance the speed of convergence.

iv) Notice that the highest number of attributes obtained from the crossover cannot exceed $i+j$. Thus repeated crossover operations on a given population whose sum total number of attributes of the true optimal chromosome is less than the chromosome length (i.e. $||Y_1| + \ldots + |Y_n| < \gamma$, where is the subset of Z_i of chromosomes that are distinct from Z_j for all $j \neq i$) will never converge to the global optimal solution. This explains why appropriate mutation operations have to be employed to make this happen. In addition, some procedures like partial reshuffling described in Section 2.3 that virtually enlarges the population size can be introduced to further enhance the convergence rate.

Observe that for a 2-point crossover, redundancy occurs 1) when all the corresponding genes in both chromosomes lying between these crossover points are identical, or 2) when all the corresponding elements lying on the end sections outside the section bounded by the two cut-points are identical. Moreover, as mentioned in i), one can make the crossover even more efficient if we only crossover those sections of the pair of chromosome having at least a prescribed number of pairs (say 2 or 3) of corresponding elements different [2,6].

2.2 The effect of mutation rate on convergence rate

The two commonly known mutation operators are the 1 bit flip mutation and the uniform mutation. The former changes the value of one of the genes chosen at random each time. If it selects a gene that needs to be mutated to the correct attribute of the optimal chromosome, then it helps bring the chromosome to be mutated closer to the optimum. If it happens to select a gene that already has the correct attribute of the optimal chromosome, then its effect will be disruptive. (i.e. it moves this chromosome farther away from the optimum.) Both the productive and disruptive effects are rather small. This mutation operator is useful especially at the final stage of reproduction, but it may not be disruptive enough to preclude the search from being trapped in a local optimal point.

The uniform mutation changes the value of every gene positioned along the length of the chromosome with a probabilityμ. When it is applied, the probability of changing any specific l positions on a binary chromosome is given by $\mu^l(1-\mu)^{\gamma-l}$. By differentiating the expression with respect to μ, it is easy to see that this probability will be maximized if $\mu = (l/\gamma)$. That is, if μ is set to be l/γ, then l bits will be most likely to be mutated in this case. Like the crossover operation, the mutation operation tends to be disruptive as well as being constructive. An appropriate rate of uniform mutation will allow those chromosomes to be mutated to have a higher probability of improving their fitness values.

For instance, when the search is close to the optimal point, most of the chromosomes in the population will be very similar to the optimal chromosome, only a few bits of changes are necessary to transform them to the optimal one. In the case of uniform mutation, one should adaptively reduce the mutation rate when this scenario occurs [3].

2.3 Partial reshuffling procedure

The partial reshuffling procedure allows the good mature chromosomes to crossover with some newly generated ones so that the genetic search can be considerably diversified [2]. This procedure allows us to perform genetic search operations effectively using a much smaller population size. The conventional way of doing this is to repeat the genetic scheme over a new randomly generated population when there is no detectable improvement in the original population after a certain number of generations. This reshuffling enlarges the search space, but it is very time consuming.

Our procedure takes advantage of the fact that a considerable number of attributes of the optimal string might have been acquired previously by some strings throughout the process of reproduction. These strings having good candidacy potential should be allowed to recombine with some new strings for further improvements. Clearly, there is a considerable time saving when compared with the usual reshuffling procedure which requires an actual restart over from the very beginning. Below we implement a form of partial reshuffling, as follows:

For a given population P of size $3n$,

1. Sort the population P by fitness, then separate the pool into two pools P_{2n} and P_n. The former contains the best 2/3 of the strings. The latter contains the remaining 1/3 of the strings.
2. Replace P_n by n randomly generated strings.
3. Generate a new pool of $3n$ strings by $\frac{3n}{2}$ crossover operations for which the first and the second parents are selected randomly from P_{2n} and P_n respectively. Keep both children in the new pool.
4. Concatenate P_{2n}, P_n and P_{3n} to form a pool of $6n$ strings. Sort this new pool and keep the best $3n$ strings. These $3n$ strings form the new population which replaces the old population P.

3 Application of the improved operators to object location

In this section, we apply the above ideas to improve the GA for the aforementioned affine matching problem. Instead of (a, b, c, d, e, f) in equation (2), select three fixed non-collinear template points and the chromosome is alternatively coded as (x_1, y_1, x_2, y_2, x_3, y_3), where (x_i, y_i), i = 1, 2, 3 are three image points that maps to the three fixed template points. Three point mapping uniquely determines an affine transformation [9].

The difference of this coding with the so called image space coding in [14] is that the three image points can be any points. It does not need to be edge points or interest points. Thus the coding is a transform space coding using the terminology of [14]. Compared with direct (a, b, c, d, e, f) coding in [14], this coding has two advantages. Firstly, there is no need to use interval arithmetic to determine the range of each parameter. Secondly, the quantization interval is not arbitrary. It is precisely the image resolution.

Let the population size be M, the crossover probability be P_c and the mutation probability be P_m. The crossover operation is as follows: for each of the x_i or y_i $(i = 1, 2, 3)$, a dice is rolled. If the result is smaller than P_c, the x_i or y_i of the two chromosomes are exchanged. Thus in a crossover operation between two chromosomes, the dice is rolled six times. The above crossover is equivalent to uniform crossover on a non-binary chromosome representation. Uniform bit mutation is used with mutation probability P_m. The following settings are used for a single GA run: $M = 100$, $P_c = 0.65$, $P_m = 0.025$. The number of generations N_g is 150. In each generation, M new chromosomes are generated. The 2M chromosomes are ranked by fitness and the M least fit chromosomes are discarded. The remaining M chromosomes form a new generation. This is a form of elitist selection. During the initialization, (x_i, y_i) are randomly selected from edge points. The fitness function $f(g)$, where g is a chromosome is defined as follows:

$$f(g) = \frac{V_g}{(D_g + 1)} \tag{4}$$

where V_g is the number of distinct points at which a transformed template point overlaps an image edge point. D_g is the average distance of a distinct transformed template point to the nearest image edge points. The distance of a point to the nearest image point is obtained from a pre-computed distance map. The value 1 is a small constant to avoid the fitness becoming infinity when D_g is zero. The fitness will be maximized when the transformed template exactly overlaps the edge points in the image.

The following schemes are tested progressively in our experiment:

i) Simple 2-point crossover without any checking for redundancy and with the crossover rate set at 0.5 and mutation rate set at 0.01. (These values are standard parameters used in the GA literature.)

ii) 2-point crossover with checking for redundancy: The 2-point crossover is performed only when the content of chromosomes is not the same. The following method is used to check the difference, if any, between two chromosomes: a) If the center part of the two chromosomes is the same, no crossover is performed; b) If the center part is not the same, but the two end parts lying outside the crossover points are the same, no crossover is performed. The algorithm will be retried up to 10 times to select a pair of new chromosomes to perform the crossover.

iii) 2-point crossover with checking and regenerate 1/3 of the chromosomes in the population for every 40 generations.

iv) 2-point crossover with checking and assurance that at least 2 elements (or genes) selected for crossover are distinct, i.e. the Hamming distance is at least 2.

v) Combinations of the above.

The purpose of these tests is to investigate the merits of the individual improved operators, as well as their relative merits.

4 Experimental results

A synthetic image – "road sign" (Figure 3a) – is used as template. Random noise at the level of 2% (3b), 5% (3c) and 8% (3d) of the whole image are added to the image. The actual number of edge points in each image is given in the figure legend. The GA achieves a success if it finds the correct solution within $\pm\varepsilon$ pixels. The results which are averages of 500 runs with $\varepsilon = 1$ are given in Table 1. The success probabilities for various schemes are shown and the corresponding number of runs N to give an overall success probability of $\geq 95\%$ is shown in brackets.

A real image edge detected using the Sobel operator is then tested. The real image, a ping pong racket, is used as the template (Figure 4a) and a database of 43 images are formed by placing various objects near the racket. 7 of these 43 images are shown in Figures 4b and 4c. These objects simulate structured noise, which are particularly difficult to deal with, as well as random noise introduced during imaging and edge detection. 500 images are used as input to the different GAs. Each image is randomly picked with equal probability from the database. The results with $\varepsilon = 2$ and 3 are given in Table 2.

The following observations are in order:

1. The probability of success for a single run of GA can be increased considerably by implementing redundancy checking, adaptive mutation and partial reshuffling. This in turn reduces the number of runs required for probability amplification. Comparing scheme 1 and 5 of the two tables, the probability of success is increased by an average factor of 7.6 (for the synthetic image) and 2.9 (for the real images). The number of runs required is reduced by an average factor of 8.8 and 2.9 respectively.
2. The most significant improvement in probability of success can be made by introducing redundancy checking (shown in **bold** in the tables). Comparing scheme 1 and 2, the probability of success is increased by an average factor of 4.3 (synthetic) and 2.2 (real) due to redundancy checking alone.
3. Redundancy checking with Hamming distance provides a significant increase in probability in one case (from 24.8 to 36.8%) but otherwise produces steady increases or even a slight decrease (from 15.8 to 14.6%).
4. The adaptive mutation (with progressively smaller mutation rate) and the partial reshuffling are useful schemes to add to redundancy checking. When introduced, they would lead to steady increase in the probability of success.

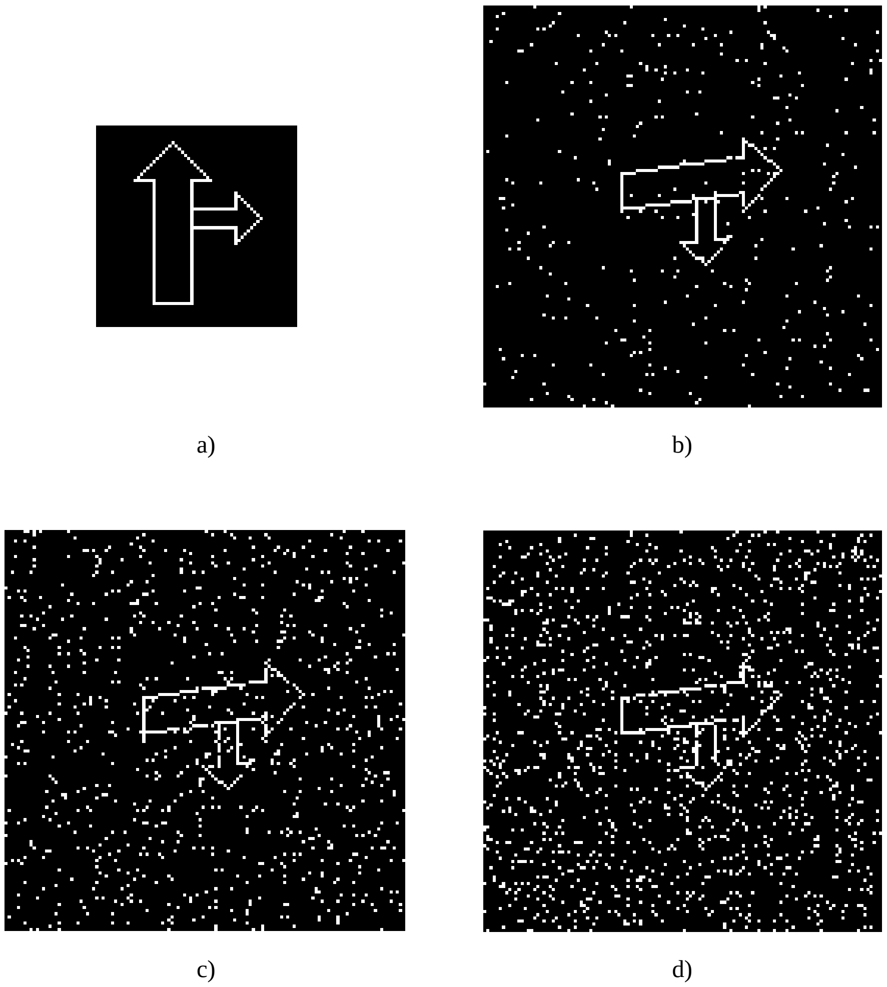
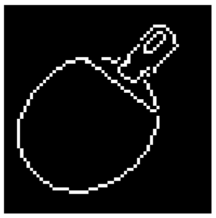
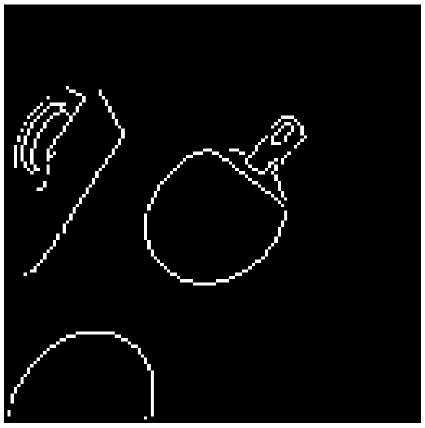


Figure 3: a) Road sign template. Before adding noise, the road sign image has 179 points. Road sign image with b) 2% random noise (total 503 points); c) with 5% random noise (total 968 points); d) with 8% random noise (total 1454 points).

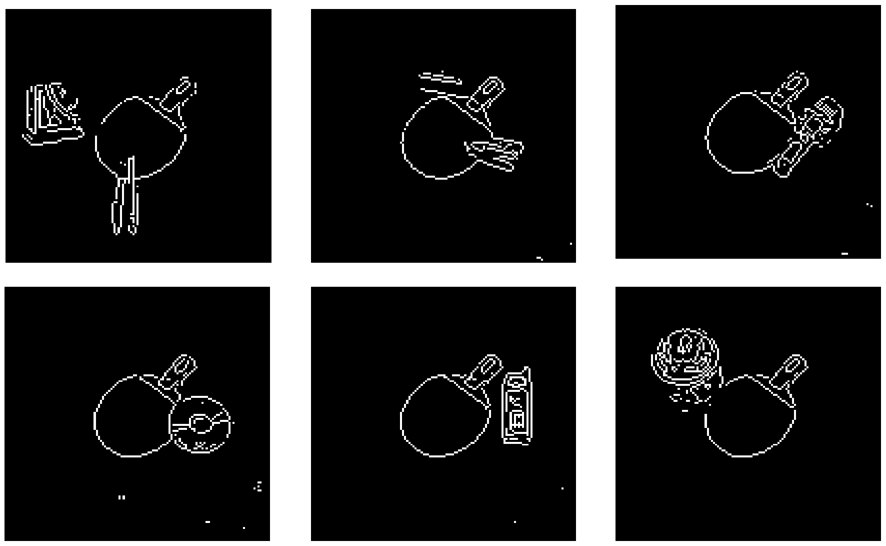
5. As control, we also evaluate the performance of 2 point crossover with addition of adaptive mutation and partial crossover individually. Comparing schemes 2, 6, and 7, introducing redundancy checking provides the greatest increase in success probability, followed by adaptive mutation and partial reshuffling.
6. We have tacitly assumed that the most consuming part of the GA is in fitness evaluations. The overhead introduced by redundancy check is small. This is true in most GAs. In particular, it is true in the affine location problem, as computing each fitness value requires transforming all the points in the template to the image.



a)



b)



c)

Figure 4: a) Real ping pong racket template (184 points). b) An edge detected image with the racket with other objects (total 404 points). c) Six other images out of a database of 43 images (shown with reduced sizes).

Table 1: Average success probability (in 500 runs) for the synthetic images

Scheme no.	Description of Method	Success probability with $\varepsilon = 1$ (number of runs to achieve 95% success)		
		2% noise	5% noise	8% noise
1	2 pts crossover only	20.4% (14)	5.0% (59)	2.8% (106)
2	**2 pts crossover with checking**	**54.8% (4)**	**24.8% (11)**	**14.8% (19)**
3	2 pts crossover with checking (at least 2 bits distinct)	56.8% (4)	36.8% (7)	17.8% (16)
4	2 pts crossover with checking + ReGen 1/3	60.2% (4)	27.6% (10)	16.2% (17)
5	2 pts crossover with checking + ReGen 1/3 + mutation (0.1-first 50 generation, 0.05- 51 to 100, 0.01-101 to 150)	69.8% (3)	43.4% (6)	30.0% (9)
6	2 pts crossover with ReGen 1/3	27%	8.2%	4%
7	2 pts crossover with mutation (0.1-first 50 generation, 0.05- 51 to 100, 0.01-101 to 150)	43%	15.8%	13.4%

Table 2: Average success probability (in 500 runs) for the real

Scheme no.	Description of Method	Success probability (number of runs required to achieve 95% success)	
		$\varepsilon = 2$	$\varepsilon = 3$
1	2 pts crossover only	6.8% (43)	9.4% (31)
2	**2 pts crossover with checking**	**15.8% (18)**	**18.6% (15)**
3	2 pts crossover with checking (at least 2 bits distinct)	14.6% (19)	19.6% (14)
4	2 pts crossover with checking + ReGen 1/3	15.4% (18)	19.6% (14)
5	2 pts crossover with checking + ReGen 1/3 + mutation (0.1-first 50 generation, 0.05- 51 to 100, 0.01-101 to 150)	22.0% (13)	23.4% (12)
6	2 pts crossover with ReGen 1/3	8%	11.6%
7	2 pts crossover with mutation (0.1-first 50 generation, 0.05- 51 to 100, 0.01-101 to 150)	13.8%	16.4%

5 Conclusions

The GA has been successfully applied to the affine object location problem. However, so far many tests have only been conducted on simple images with no noise [9,10,14]. For complicated images with random and structured noises, the performance of the GA may degrade drastically. To tackle this problem, a repeated genetic algorithm [12] may be used; when the probability of success of a single copy of GA is not critically small, one can always have a guarantee of a high overall probability of success after several runs of the GA. The idea is known as probability amplification. Since the amplification relationship is exponential, when the probability of success is small, i.e. the problem is difficult, a small increase in the probability of success can often significantly decrease the number of runs. Thus it is worthwhile to investigate GAs with improved operators and better probability of success. In this paper, we investigated three improved GA operators on the affine object location problem under large random and structured noises . We found that redundancy checking, adaptive mutation and partial reshuffling are generally good heuristics and give an intuitive explanation of why these ideas are worthy by examining the working principles of the GA. The experimental results suggest that redundancy checking give the most significant improvement.

One explanation can be found by examining the classical no free lunch theorems (NFL) [11], which states that the average performance of *all* (stochastic or deterministic) algorithms is equal when averaged over *all* possible function landscapes, when the algorithm does not visit a search location more than once. Thus redundancy checking is guaranteed to improve the GA probability of success. This in turn suggests that redundancy checking should be more widely applied in future GA designs for vision problems.

References

[1] L. Chambers, *The Practical Handbook of Genetic Algorithms: Applications*, 2^{nd} Ed., Chapman & Hall, 2001.

[2] B.K.S. Cheung, A. Langevin, and B. Villeneuve, "High Performing Evolutionary Techniques for Solving Complex Location Problems in Industrial System Design", *Journal of Intelligent Manufacturing*, 12, 455–466, 2001.

[3] A.E. Eiben, R. Hinterding, Z. Michalewicz, "Parametic Control in Evolutionary Algorithms", *IEEE Trans. Evolutionary Computation*, 3(2), 124–141, 1999.

[4] D. Greenhalgh, S. Marshall, "Convergence Criteria for Genetic Algorithms", *SIAM J. Comput.*, 30(1), 269–282, 2000.

[5] J. He, X. Yao, "Towards An Analytic Framework for Analysing The Computation Time of Evolutionary Algorithms", *Artificial Intelligence*, 145, 59–97, 2003.

[6] S. Ronald, "Duplicate Genotypes in A Genetic Algorithm", *Proc. IEEE Int. Conf. Evolutionary Computation*, IEEE Press, Piscataway, NJ, 793–798, 1998.

[7] W.J. Rucklidge, "Efficiently Locating Objects Using The Hausdorff Distance", *International Journal of Computer Vision*, 24(3), 251–270, 1997.

[8] G. Rudolph, "Finite Markov Chain Results in Evolutionary Computation: A Tour d'Horizon", *Fundamenta Informaticae*, 35, 67–89, 1998.

[9] P.W.M. Tsang, "A Genetic Algorithm for Aligning Object Shapes", *Image and Vision Computing*, 15, 819–831, 1997.

[10] P.W.M. Tsang, "Enhancement of A Genetic Algorithm for Affine Invariant Planar Object Shape Matching Using the Migrant Principle", *IEE Proc. Vision, Image & Signal Process*, 150, 107–113, 2003.

[11] D.H. Wolpert and W.G. Macready, "No Free Lunch Theorems for Optimization", *IEEE Trans. on Evolutionary Computation*, 1, 67–82, 1997.

[12] S.Y. Yuen, C.K. Fong and H.S. Lam "Guaranteeing The Probability of Success Using Repeated Runs of Genetic Algorithm", *Image and Vision Computing*, 19, 551–560, 2001.

[13] D.A. Forsyth, J. Ponce, *Computer Vision, a modern approach*, Prentice Hall, 2003.

[14] G. Bebis, S. Louis, Y. Varol, A. Yfantis, "Genetic Object Recognition Using Combinations of Views", *IEEE Trans. Evolutionary Computation*, 6(2), 132–146, 2002.

[15] P. Vitányi, "A Discipline of Evolutionary Programming", *Theoretical Computer Science*, 241, 3–23, 2000.

[16] B.K.S. Cheung, S.Y. Yuen, C.K. Fong, "Enhancement in Performance of Genetic Algorithm for Object Location Problem", *Proc. 8th International Conference on Control, Automation, Robotics and Vision*, Kumming, China, 692–697, Dec. 2004.