Available online at www.sciencedirect.com
SCIENCE DIRECT
Image and Vision Computing 23 (2005) 972–980
image AND vision COMPUTING
www.elsevier.com/locate/imavis

A robust iterative hypothesis testing design of the repeated genetic algorithm

Shiu Yin Yuen[a,*], Hoi Shan Lam[b], Chun Ki Fong[b], Shi Feng Chen[a], Chi Kin Chow[a]

[a]*Department of Computer Engineering and Information Technology, City University of Hong Kong, Kowloon Tong, Hong Kong, Peoples' Republic of China*
[b]*Department of Electronic Engineering, City University of Hong Kong, Kowloon Tong, Hong Kong, Peoples' Republic of China*

Received 30 January 2004; received in revised form 27 May 2005; accepted 1 July 2005

Abstract

The genetic algorithm is a simple and interesting optimization method for a wide variety of computer vision problems. However, its performance is often brittle and degrades drastically with increasing input problem complexity. While this problem is difficult to overcome due to the stochastic nature of the algorithm, this paper shows that a robust statistical design using sequential sampling, repeated independent trials and hypothesis testing can be used to greatly alleviate the degradation. The working principle is as follows: The probability of success P of a stochastic algorithm A (in this case A is the genetic algorithm) can be estimated by running N copies of A simultaneously or running A repeatedly N times. Such a scheme is generally referred to as the parallel or repeated (genetic) algorithm. By hypothesis testing, P can be tested with a required figure of merit (i.e. the level of significance). This is used in turn to adjust N in an iterative scheme to maintain a constant P_{repeated}, achieving a robust feedback loop. Experimental results on both synthetic and real images are reported on the application of this novel algorithm to an affine object detection problem and a free form 3D object registration problem.

Keywords: Repeated genetic algorithm; Probability of success; Free form object registration

1. Introduction

Genetic Algorithm (GA) is an interesting stochastic optimization technique that is directly inspired by analogy with genetics and evolution theories. In marked contrast to serial optimization techniques such as the gradient ascent, the GA employs a population of potential solutions, known as chromosomes. A set of M chromosomes forms the current generation. The next generation of chromosomes is obtained by modifying the chromosomes by three operators: selection, followed by crossover and mutation. The procedure of the GA can be summarized below:

1. Randomly initialize a population of M chromosomes.
2. Select two chromosomes for mating from the population.
3. Perform crossover with probability P_c to obtain two offspring chromosomes.
4. Perform mutation with probability P_m to the offspring chromosomes, calculate their fitness and then add them to a new population.
5. Repeat step 2–4 until the new population reaches the same size as the current population.

In spite of promising successes in the application of the GA to computer vision (e.g. [1–4]), its performance is often brittle and degrades with increasing problem complexity. Owing to the fact that it is merely an intelligent random search algorithm, there can be no guarantee that it has found the correct solution (i.e. the global maximum) given an unknown problem landscape. Theoretical work on the GA's probability of success [5,6] has been encouraging, but has not advanced to a stage ready for practical application, especially in the case of unknown problem class (unknown complexity). The GA's probability of success P for a particular landmark test image *supplied by the researcher* can be obtained by running it on the same image N times, then takes the number of times N_{success} of which the GA successfully finds the correct solution. In this case N_{success}/N

* Corresponding author. Tel.: +86 852 27887717; fax: +86 852 27888292.
E-mail address: itkelvin@cityu.edu.hk (S.Y. Yuen).

0262-8856/$ - see front matter © 2005 Elsevier B.V. All rights reserved.
doi:10.1016/j.imavis.2005.07.001

approaches P as N approaches infinity by the law of large numbers. This, along with other statistical figures such as the standard deviation, is usually the way to justify that the GA works well in a particular application in previous researches. However, this figure of merit P has no bearing on the actual probability of success of a totally new and unknown image. In fact, the GA's performance is often brittle, failing completely when given a new, unknown image of higher complexity than the landmark test image.

Whilst this problem cannot be completely overcome due to the stochastic nature of the algorithm, this paper shows that a sequential sampling design [7] using repeated independent trials and hypothesis testing can be used to design a more robust algorithm to input images of varying complexities. The idea is to test online the probability of success P, then modify N to meet the requirement on the overall probability of success P_{repeated} for N applications of the GA in an iterative feedback loop.

Section 2 introduces the repeated GA (RGA) [8] and our iterative hypothesis testing (IHT) method for increasing the robustness of the RGA. Section 3 reports the experimental results on applying the IHT-RGA to an affine object detection problem and a free form 3D object registration problem. Section 4 gives a conclusion. A preliminary version of this paper has appeared in [9].

2. Repeated genetic algorithm

Suppose instead of a single run of the GA, the algorithm runs the GA N ($N>1$) times on the same image. Each run is kept independent, with no information passed between each run. At the end of each run, the best solution found is recorded. Then the chromosomes are randomly initialized to begin the next run. Such an arrangement is known as the repeated GA (RGA). It is a special case of the parallel GA in the literature [10].

Let the probability of success of a single run be P_{SGA}. It is the probability that the best solution found in a single run is the correct (i.e. globally maximum) solution. Let the probability of success after running N times be P_{RGA}. It is the probability that any one of the N best solutions is the correct solution. Since the N runs are independent,

$$P_{\text{RGA}} = 1-(1-P_{\text{SGA}})^N \tag{1}$$

Given a user specified figure of merit P_{RGA} (e.g. $P_{\text{RGA}}\geq 0.95$), the required number of runs N is simply

$$N \geq \frac{\ln(1-P_{\text{RGA}})}{\ln(1-P_{\text{SGA}})} \tag{2}$$

Thus provided that P_{SGA} is known, one can guarantee the RGA's overall probability of success. This is a general technique: Given any stochastic optimization method with probability of success P_{S}, one can amplify its probability of success to P_{R} by applying it independently N times. This is known as *probability amplification* or probability boasting [11,12]. Given a fixed P_{S}, the probability of not finding the optimal solution decreases exponentially with N.

In [8], we report an estimation method for P_{SGA} from a set of training images. Using the procedure described in [8], the lower confidence bound and the confidence level can be estimated (see [13] for the exact definitions; the level of significance in [8] should in fact be the confidence level). In this paper, we explore another route, which is to estimate P_{SGA} from N runs on a *single* input image. The reasoning is as follows:

Suppose s is the best solution amongst the N solutions. If it is assumed that it is the correct solution, then on average it appears around $N\times P_{\text{SGA}}$ times amongst the N solutions. Thus if the assumption is correct, then from the actual number of times that s appears, we can say something about P_{SGA}.

Let c_1 be the number of times s appears in the N runs. To test that P_{SGA} is larger than some lower bound P_{est}, the following hypothesis test may be used:

$$H_0 : P_{\text{SGA}} = P_{\text{est}} \qquad H_1 : P_{\text{SGA}} > P_{\text{est}} \tag{3a}$$

Using Eq. (1), one estimates P_{est} to be

$$P_{\text{est}} = 1-e^{\frac{\ln(1-P_{\text{RGA}})}{N}} \tag{3b}$$

and set up a fixed hypothesis test for each N a priori.

Suppose N trials have been conducted. Then the level of significance of the corresponding hypothesis test can be computed. If it is smaller than the user specified value α, then the algorithm exits with success. Otherwise N is increased by 1 and the algorithm iterates unless $N>N_{\text{limit}}$, in which case the algorithm exits with failure.

The complete algorithm is summarized below:

Iterative Hypothesis Testing (IHT) RGA

Input: A single input I, α, P_{RGA}, N_{init}, N_{limit}.

1. Set $N=N_{\text{init}}$. Run RGA.
2. Find the best solution s amongst the N runs and the number of times c_1 that it occurs.
3. Compute the level of significance of the corresponding hypothesis test (Eq. (3)).
4. If the level of significance is smaller than or equal to α, exit with success.
5. Run the GA one more time, i.e., $N\leftarrow N+1$.
6. If $N\leq N_{\text{limit}}$, go to step 2, otherwise exit with failure.

Output: Success/failure flag, best solution s.

Let give an illustrative example. Let set $\alpha=0.01$, $P_{\text{RGA}}=0.95$, $N_{\text{init}}=5$, $N_{\text{limit}}=\infty$. Initially, five GA runs are made. Suppose the solutions found in the five runs are $s_1,\ldots,s_5$. Suppose the fitness values are $f(s_1)=f(s_4)>f(s_2)>f(s_3)>f(s_5)$ and s_1 and s_4 are identical solutions.[1]

[1] This condition can be relaxed. In the case where there is more than one global optimum with the same fitness value, we only require $f(s_1)=f(s_4)$ and do not require s_1 and s_4 to be identical. The probability of success is then the probability that one of the global optima is found.

Then $s=s_1=s_4$ is the best solution amongst the five runs, and it occurs in $c_1=2$ out of $N=5$ runs (step 2), which gives a level of significance of 0.0674 (step 3). Since it is larger than α, it leads to an informed guess that s may not be the correct solution (step 4). So the GA is run one more time and the situation is evaluated again (step 5).

Note that N_{init} cannot be 1, otherwise the IHT algorithm will exit with success trivially. In practice, N_{init} may be set to any reasonable value larger than 1, or better still, from a priori knowledge about the expected complexity of the input image [8].

On the other hand, N_{limit} is a threshold determined by the maximum amount of computational resources that the algorithm is prepared to expend on finding the solution. A clear physical meaning of N_{limit} can be seen from the following equation, obtained by rearranging Eq. (1):

$$P_{\text{SGA limit}} \geq 1-(1-P_{\text{RGA}})^{\frac{1}{N_{\text{limit}}}} \tag{4}$$

Thus N_{limit} gives the minimum probability of success for a single run of GA on an input image. In other words, it represents a threshold probability below which the engineer prefers not to determine the best solution, for example, because of lack of time or computing resources. Another reason may be because finding the best solution with such a low probability has little importance in a particular application.

A major advantage of this RGA algorithm is that it is able to automatically adjust the number of runs N. For easy images, N shall be small whereas N shall be large for complex images. This automatic adjustment is done *without* knowing a priori the complexity of the image. Thus the algorithm is more robust. This desirable characteristic is lacking in many existing GA implementations. These implementations are typically brittle, working well with some fairly complicated images but may not work as well with other images.

The algorithm exploits the necessary condition that the probability of success of a single run P_{SGA} must satisfy. We hasten to remark that the condition is not sufficient, since it is impossible to guarantee that the best solution s found from the N runs is indeed the correct (i.e. globally maximum) solution. In fact, *no* stochastic algorithm can guarantee that such is the case unless the search is so exhaustive that it tests the entire search space once (for example, consider applying any stochastic algorithm on a flat landscape and a single peak).

Thus the algorithm cannot guarantee that it has found the correct solution if it exits with success. Paradoxically, if it exits with failure, then it can guarantee that the best solution found has less than a probability of P_{RGA} of being the correct solution, and the level of significance of this statement is α. Thus if it fails to find the correct solution, it can detect the failure confidently.

The basic form of the algorithm is reminiscent of sequential sampling (SS) [7] in statistics. In SS, multiple samples, with possibly different sample sizes, are taken until a fixed hypothesis can be definitely accepted or rejected. The above algorithm looks alike a SS with a fixed sample size of 1. However, despite the similarity, intrinsically, the IHT algorithm does not test a fixed hypothesis. When the algorithm runs one more time, it may discover a better solution than its current best solution. (Please refer to step 2 in the IHT algorithm above). When this is the case, a new hypothesis is formed, and the previous statistics become valueless. Also our application of the method in optimization problems has a different semantics to the conventional usage of SS in statistics. The Iterative Hypothesis Testing is a more appropriate name to reflect the nature of the method.

In the RGA approach, each GA run evolves independently. This can be contrasted with the niching approach, where the population pool of a single GA is divided into several subpopulations, each of which is called a species. Each species attempt to find its own niche, which is a disjoint problem subspace. In the niching approaches, different species interact through competition, voting or fitness sharing. Some examples of niching can be found in [4,14], which also contain a review of other niching approaches. The use of niching may not be beneficial in the RGA. Firstly, communication between subpopulations has potential drawbacks such as the reduction of diversity of solutions. There is no proof that allowing such communication will definitely increase the probability of success for individual runs in all applications. Secondly, some niching methods require control parameters and it is highly unlikely that a set of control parameters exist that are good for all applications. Thirdly, though niching methods that do not introduce additional parameters have been proposed [4], our own experience with [4] suggests that the performance is problem dependent.

When problem knowledge is available, one can design a good GA specifically for the problem. For example, the nonlinear crossover is especially useful for the traveling salesman problem [15]. Using such improved GAs will increase the probability of success of a single run P_{SGA}. This is significant, as in equation (2), N will decrease exponentially as P_{SGA} increases. Then one can use the probability amplification and the IHT framework to deal with unknown problem complexity within the same problem class.

3. Applications

3.1. Application to affine object location

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 [16] and tree search [17] 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.

The GA has been applied to solve the object recognition problem due to viewpoint changes [3,18]. 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 problem:

Given a template $T=\{(x,y)\}$ and an image $I=\{(x', y')\}$ containing a single instance of an affine transformed copy of T, the problem is to recover the unknown affine transformation (a,b,c,d,e,f), where

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} a & b \\ c & d \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} e \\ f \end{bmatrix} \tag{5}$$

In this paper, the novel method is applied to the above object location problem.

Instead of (a,b,c,d,e,f), 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 [3]. The difference of this coding with the so called image space coding in [18] 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 [18]. Compared with direct (a,b,c,d,e,f) coding in [18], 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:

$$f(g) = \frac{V_g}{(D_g + 1)} \tag{6}$$

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 settings are used for the IHT: $P_{RGA}=0.95$. $N_{init}=4$, $N_{limit}=\infty$. In actual implementation, the GA is hybridized with local search: It is considered a success if the GA finds the solution within ± 1 pixels. This leads to a slightly more complicated procedure for determining the best solution in the IHT loop, but the basic idea is the same. After the RGA, a local search is used to find the most accurate solution.

The input images are 128×128 in size. The following parameter restrictions are imposed on the images:

$$0.8 \le \mathrm{abs}\left(\begin{vmatrix} a & b \\ c & d \end{vmatrix}\right) \le 1.0 \quad -1 \le a,b,c,d \le 1 \tag{7}$$

This is equivalent to restricting the scaling down of the template due to the affine transformation, that is, the object cannot be too small. It prevents the GA from converging on small patches of random edge points. Chromosomes whose affine transform does not satisfy Eq. (7) will be assigned a fitness of 0.

To investigate the performance of the IHT-RGA on images of varying complexities, 100 images of a pair of scissors generated by random affine transformations with 0, 2, and 5% (of the whole image) added random noise are used as input (Fig. 1).

The Q factor [8] measures the speedup over an exhaustive search:

$$Q = \frac{P_{RGA}}{P_{norm}} \tag{8}$$

where $P_{norm}=t_{RGA}/t_{es}$ is the proportion of search probes that has been performed. t_{RGA} is the total number of search probes by the RGA and t_{es} is the total number of possibilities in the search space.

The results for $\alpha=0.1$ are shown in Table 1. The average N for the 0, 2 and 5% noise levels are 6.14, 48 and 67.87; the actual success rates are 97, 89 and 80%; the average Q factors are 114,226, 14,712 and 10,408, respectively.

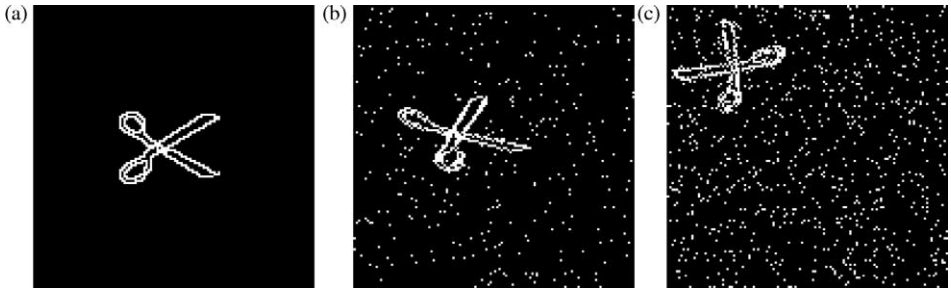

Fig. 1. Synthetic scissors image with (a) 0%, (b) 2% and (c) 5% noise.

The results for $\alpha=0.01$ are shown in Table 2. The average N for the 0, 2 and 5% noise levels are 9.68, 66.39 and 122.58; the actual success rates are 99, 92 and 90%; the average Q factors are 72,662, 10,640 and 5765, respectively.

It is clearly shown that the algorithm can automatically adjust N in proportion to the input complexity without knowing the input complexity a priori. The algorithm increases the number of runs *automatically* for difficult scenes.

Though the hypothesis testing is only a necessary but not sufficient condition for finding the solution with $P_{\mathrm{RGA}}=95\%+$, the average probability of success is 88.7% (for $\alpha=0.1$) and 93.7% (for $\alpha=0.01$). This indicates that the algorithm is working robustly against scenes of various complexities, *without knowing the complexity a priori*. It is interesting since it automatically adjusts itself when presented with difficult scenes to maintain a fairly constant level of probability of success. It is also clear that a smaller α will give a better average probability of success. However, the average N required is also larger, which gives a smaller average Q factor.

A control experiment is performed: when $\alpha=0.1$, the average N for 0% noise image is 6.14. If a fixed $N=7$ is used, then for 2% noise images, the success rate is 30%, for 5% noise images, the success rate is 20%. When $\alpha=0.01$, the average N for 0% noise images is 9.68. If a fixed $N=10$ is used, then for 2% noise images, the success rate is 43%, for 5% noise images, the success rate is 28%. This shows clearly that both the GA and the RGA designs with a fixed N are a lot more brittle.

From the Q factors, it is shown that the algorithm takes more time to analyze complex scenes. Human beings display the same phenomenon (e.g. the Dalmatian dog image [19]). Refer to Table 2. For $\alpha=0.01$ and 5% noise, the maximum N is 1011, which corresponds to a Q factor of 699. This shows that the IHT-RGA has a hard time working with *that particular* 5% noise level image. In this situation, if we have had set the N_{limit} to some smaller limiting value, the IHT would know the particular run is problematic. This control signal may be used to trigger a more sophisticated vision algorithm. Thus this robust algorithm also has the ability to detect that it is not working properly. Note that this is a very desirable characteristic.

A second experiment was performed on another synthetic image. For 0% noise image, the success rates are 99%. The minimum and maximum N are 4 and 13. For 5% noise image, the respective figures are 94%, 35, and 1352. Since the results are similar, the details are omitted.

Experiments have also been performed for real images. A real image is a lot harder to deal with since after edge detection, the image is degraded by edge localization errors, random noise and missing boundaries. A template that is a racket (Fig. 2a) is used. Structured noise is introduced by putting various objects beside the racket. Forty-three real images which have been edge detected are used. Fig. 2b shows some samples of the images. Since the edge contours suffer from distortions due to imaging, digitization and localization errors, it is considered a success if the GA finds the solution within ± 4 pixels. The other settings of the GA remain unchanged.

The IHT algorithm is run once on each of the 43 images using $\alpha=0.01$. The algorithm correctly locates the racket in all the 43 images, despite the different complexities of the images. The maximum, minimum and average N are 1354, 17 and 132.7, respectively. This

Table 1
Results of affine object location for $\alpha=0.1$

	0% Noise	2% Noise	5% Noise
Max N	19	231	342
Min N	4	4	4
Average N	6.14	48	67.87
Max time (second)	41	422	765
Min time (second)	7	7	7
Average time (second)	13.9	88.1	134.5
Success rate	97%	89%	80%

Table 2
Results for affine object location for $\alpha=0.1$

	0% Noise	2% Noise	5% Noise
Max N	35	298	1011
Min N	4	4	4
Average N	9.68	66.39	122.58
Max time (second)	84	575	1826
Min time (second)	8	7	8
Average time (second)	21.58	127.63	223.2
Success rate	99%	92%	90%

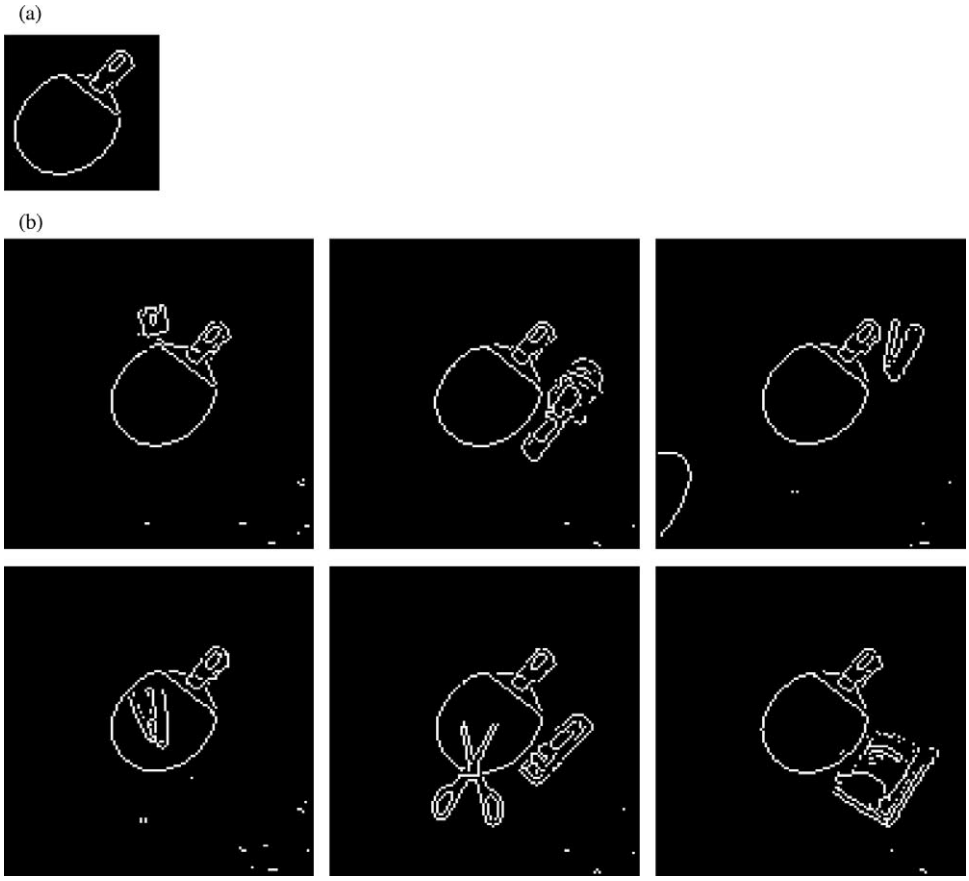

Fig. 2. Real images: (a) The racket template; (b) edge detected images of the racket with other objects.

experiment demonstrates the robustness of the algorithm in real images.

3.2. Application to free form registration

In this experiment, the proposed method is tested on a free form surface registration problem. An accurate, robust and fast registration solution is important in medical image processing and reconstruction. Many registration algorithms have been proposed in recent years. They are mainly divided into two classes: Correspondence matching [20] and Iterative Closest Point (ICP) algorithm [21]. In the correspondence matching approach, correspondences are extracted with some feature extraction processes. The selected features for matching, such as points, line, edges and regions, should be invariant to the movement of the object. However, it is well known that the correspondence problem is not easy to solve as the number of features become large. Moreover, there is no unique feature that can represent all 3D objects. Hence, the approach of correspondence matching is highly application dependent. Although the correspondences can be marked by the user, this process is very time consuming and automatic surface registration is therefore difficult. For the ICP-based algorithms, a good initial guess is necessary in order to find the correct solution. If the initial guess is far from the actual solution, incorrect solution or mismatching will happen no matter how many times the algorithm is processed. By formulating surface registration as an optimization problem [22], the registration procedure does not depend on a good initial guess, nor require prior information on correspondences or feature points given by the users.

Given a set of points $\{P_i\}$ in S_1 (size N_1) and $\{Q_j\}$ in S_2 (size N_2), they are well registered if the objective function (9) is minimized:

$$F(T) = \underset{33\%}{\text{Med}}\{E_i\} \text{ for } 1 \le i \le N_1 \tag{9}$$

where T is the transformation,

$\text{Med}_{33\%}$ is the median measurement with a 33% breakdown point (i.e. 33% overlap),

$$E_i = \min_j |H(P_i, T) - Q_j| \tag{10}$$

$H(P_iT)$ is the transformed P_i with transformation T and $T_{\min}$ is the resultant transformation from S_1 to S_2.

$$T_{\min} = \arg\min_{\mathrm{T}} F(T) \tag{11}$$

We represent the geometric transformation T between two surfaces as a chromosome that consists of six parameters: T_x, T_y and T_z are the translation genes on x, y, and z axis; α, β, θ are the rotation genes about the x, y and z

axis respectively.

$$T = SR_xR_yR_z \tag{12}$$

where

$$R_x = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\alpha & \sin\alpha & 0 \\ 0 & -\sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$$R_y = \begin{pmatrix} \cos\beta & 0 & -\sin\beta & 0 \\ 0 & 1 & 0 & 0 \\ \sin\beta & 0 & \cos\beta & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$$R_z = \begin{pmatrix} \cos\theta & \sin\theta & 0 & 0 \\ -\sin\theta & \cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \quad S = \begin{pmatrix} 1 & 0 & 0 & T_x \\ 0 & 1 & 0 & T_y \\ 0 & 0 & 1 & T_z \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{13}$$

The two surfaces to be registered in this experiment are generated from a synthetic human heart model obtained from [23]. These are range data taken from different directions with various percentages of overlap. Generally, the percentage of overlap is unknown. The corresponding rendered images with approximately 50% overlap are shown in Fig. 3. To investigate the performance of IHT-RGA on surfaces with various percentages of overlap, the registration process is repeated with two input surfaces shown at Fig. 4, which have approximately 40 and 33% overlap with the surface shown at Fig. 3(a).

The population pool size is 100. In each generation of a single GA run, 100 chromosomes are generated by 3-point crossover together with single-point mutation. 50 chromosomes with best fitness are selected from these 100 chromosomes. Another 100 chromosomes are generated by single-point mutation only. Fifty chromosomes with best fitness are selected from the mixture of the current generation and the mutation-only generated chromosomes. The GA is terminated if the rate of improvement within 10 consecutive generations is smaller than 5%. For detailed GA design, please refer to [22]. The following settings are used for the IHT: $P_{\mathrm{RGA}}=0.95$, $N_{\mathrm{init}}=2$, $N_{\mathrm{limit}}=\infty$ and $\alpha=0.1$. The registration processes are performed on a PC with Pentium IV 1.7 GHz CPU and 512 MB memory.

As seen from Table 3, the average N for overlap at 50, 40 and 33% are 4, 7.2 and 13.4; the success rate of IHT are 97, 91 and 86%. As a control experiment, the actual success rates of four independent GA run are 96, 83 and 67%, respectively.

Similar conclusions can be drawn from this experiment. It confirms the prowess of the IHT-RGA on the free form registration problem.

4. Conclusions

The genetic algorithm is an interesting optimization method for a wide variety of engineering problems. However, its performance is often brittle and degrades drastically with increasing input complexity. Since the complexity of an input image is in general not known a priori, this seriously limits the applicability of the GA as a general tool. While this problem is difficult to overcome due to the stochastic nature of the algorithm, this paper shows that a sequential sampling design using repeated GA trials and hypothesis testing can be used to design an iterative (feedback) algorithm that is adaptive and substantially more robust. Experimental results are reported on the application of the algorithm to an object detection problem and a free form 3D object registration problem. The results confirm its prowess and demonstrate three interesting characteristics: (1) the proposed algorithm works robustly against scenes of

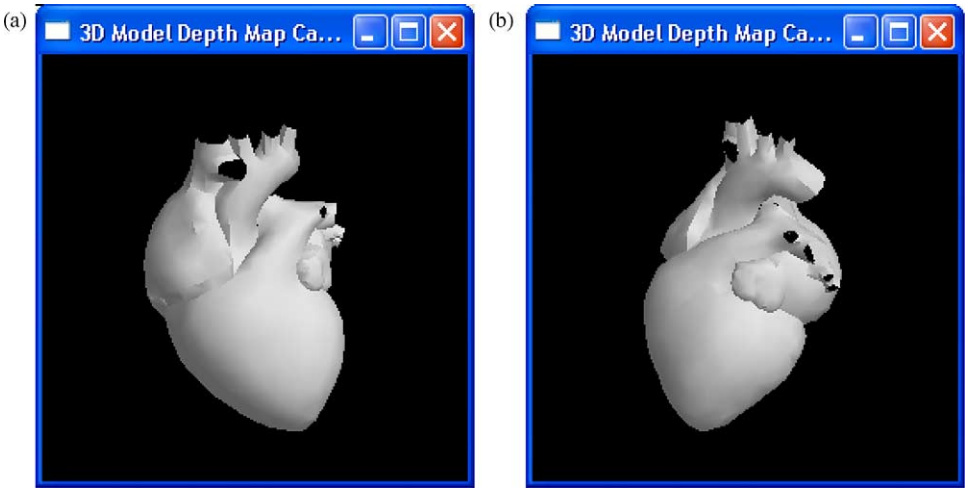

Fig. 3. Range data of two surfaces. Surface (b) has 50% overlap with surface (a). Seen from above, surface (b) has been rotated clockwise.

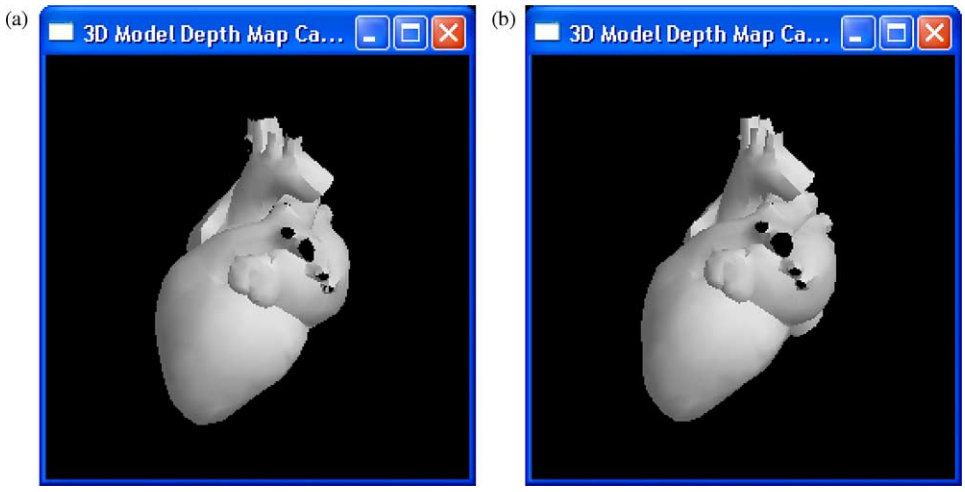

Fig. 4. Range data of surfaces to be registered which have (a) 40% and (b) 33% overlap with Fig. 3(a).

various complexities, *without knowing the complexity a priori*; (2) it is able to detect that it is not working properly and (3) the level of significance serves as a figure of merit for the quality of the solution.

Note that the statistical design can equally be applied to other stochastic algorithms to increase their robustness, since it effectively treats the GA as a black box. The IHT-RXA, where X is any optimization problem, is a topic for further investigation.

Finally, the IHT-RGA design relates the classical statistical technique of sequential sampling with optimization problems.

4.1. Post note

One might wonder whether it is possible to design a better stochastic algorithm A that is capable of giving an estimate of its own probability of success P. The answer is negative for all search algorithms, deterministic or stochastic, that samples the search space, except in the trivial sense of $P=P_{\text{norm}}$ when t_{A} only counts non-redundant visits. The argument is as follows: A (deterministic or stochastic) algorithm that samples the search space may be defined as a rule to visit the space non-redundantly in a certain order [24]. Thus one may always contrive a search space that puts the global optimum at the end of the visit. It follows that no matter what the algorithm is, the probability of success is still $P=P_{\text{norm}}$. Because of the above, it does not appear useful to derive a necessary and sufficient condition for the probability of success in the above work.

However, this state of affairs can be improved upon if *landscape assumptions* are made. In [5], time complexity expressions are given for several simple landscape classes. In [8], a method is given for an implicit landscape class defined by a working domain. In fact, different landscape assumptions are behind the efficiency of every meta-heuristic. These assumptions are currently not well defined.

Table 3
Results for free from surface registration for $\alpha=0.1$

Overlap	50%	40%	33%
Max N	6	10	18
Min N	3	4	6
Average N	4	7.2	13.4
Max time (second)	69	108	192
Min time (second)	30	48	69
Average time (second)	45	83	146
Success rate	97%	91%	86%

Acknowledgements

We thank Dr. H.P. Lo for good statistical advice, Dr. Bernard Cheung for helpful discussions and two anonymous referees for constructive comments and suggestions. The work described in this paper was supported by a grant from the Research Grants Council of the Hong Kong Special Administrative Region, China [Project No. CityU 1157/00E]. C.K. Fong was supported by a Research Studentship from the City University of Hong Kong. S.F. Chen was supported by a Research Studentship funded by a grant from the Research Grants Council of the Hong Kong Special Administrative Region, China.

References

[1] A. Hill, C.J. Taylor, Model-based image interpretation using genetic algorithms, Image and Vision Computing 10 (5) (1992) 295–300.
[2] G. Roth, M.D. Levine, Geometric primitive extraction using a genetic algorithm, IEEE Transaction on Pattern Analysis and Machine Intelligence 16 (9) (1994) 901–905.

[3] W.M. Tsang, A genetic algorithm for aligning object shapes, Image and Vision Computing 15 (1997) 819–831.
[4] S.Y. Yuen, C.H. Ma, Genetic algorithm with competitive image labelling and least square, Pattern Recognition 33 (12) (2000) 1949–1966.
[5] J. He, X. Yao, Towards an analytic framework for analysing the computation time of evolutionary algorithms, Artificial Intelligence 145 (2003) 59–97.
[6] S.Y. Yuen, Cheung, B.K.S., Bounds for probability of success of classical genetic algorithm based on hamming distance. IEEE Transactions on Evolutionary Computation.
[7] E.L. Grant, R.S. Leavenworth, Statistical Quality Control 1988.
[8] S.Y. Yuen, C.K. Fong, H.S. Lam, Guaranteeing the probability of success using repeated runs of genetic algorithm, Image and Vision Computing 19 (2001) 551–560.
[9] S.Y. Yuen, H.S. Lam, C.K. Fong, A Novel Robust Statistical Design of The Repeated Genetic Algorithm LNCS 2124 2001 pp. 668–675.
[10] E. Cantu-Paz, D.E. Goldberg, Efficient parallel genetic algorithms: theory and practice, Computer Methods in Applied Mechanics and Engineering 186 (2000) 221–238.
[11] P. Vitányi, A Discipline of evolutionary programming, Theoretical Computer Science 241 (2000) 3–23.
[12] G. Rudolph, Finite Markov chain results in evolutionary computation: a tour d'horizon, Fundamenta Informaticae 35 (1998) 67–89.
[13] E.L. Lehmann, Testing Statistical Hypotheses, second ed., Wiley, 1986. pp. 89–94.
[14] J.P. Li, M.E. Balazs, G.T. Parks, P.J. Clarkson, A species conserving genetic algorithm for multimodal function optimization, Evolutionary Computation 10 (3) (2002) 207–234.
[15] C.R. Reeves, J.E. Rowe, Genetic Algorithms—Principles and Perspectives, A Guide to GA Theory, Kluwer Academic, 2003.
[16] D.A. Forsyth, J. Ponce, Computer Vision, A Modern Approach, Pearson, 2003.
[17] W.J. Rucklidge, Efficiently locating objects using the Hausdorff distance, International Journal of Computer Vision 24 (3) (1997) 251–270.
[18] G. Bebis, S. Louis, Y. Varol, A. Yfantis, Genetic object recognition using combinations of views, IEEE Trans, Evolutionary Computation 6 (2) (2002) 132–146.
[19] R. Sekuler, R. Blake, Perception, second ed. 1990 p. 129.
[20] S.M. Yamany, A.A. Farag, Free-Form Surface Registration Using Surface Signatures Proceedings of 7th IEEE International Conference on Computer Vision, vol. 2, 1999 pp. 1098–1104.
[21] T. Masuda, K. Sakaue, N. Yokoya, Registration and Integration of Multiple Range Images for 3-D Model Construction Proceedings of 13th International Conference on Pattern Recognition, vol. 1, 1996 pp. 879–883.
[22] C.K. Chow, H.T. Tsui, T. Lee, Surface registration using a dynamic genetic algorithm, Pattern Recognition 37 (1) (2004) 105–117.
[23] http://www.3dcafe.com/asp/meshes.asp.
[24] D.H. Wolpert, W.G. Macready, No free lunch theorem for optimization, IEEE Transactions on Evolutionary Computation 1 (1) (1997) 67–82.