--

--

For CISST'98 Conference Proceedings

(Bebis et. al.)

-- --

Using Genetic Algorithms for Model-Based Object Recognition

George Bebis, Sushil Louis and Yaakov Varol

Department of Computer Science
University of Nevada
Reno, NV 89557
bebis@cs.unr.edu

Abstract

We investigate the application of genetic algorithms for recognizing real, planar, objects from two-dimensional intensity images, assuming that the viewpoint is arbitrary. Given an object and an unknown scene, the goal of object recognition is to find a geometric transformation that brings subsets of features, making up the object, and subsets of features, comprising the scene, into alignment. In general, this problem can be solved through an exhaustive search in one of the following two spaces: the space of all possible matches between model feature and scene features ("image space") or the space of all possible transformations ("transformation space"). In this paper, we propose using genetic algorithms for searching these spaces efficiently. Genetic algorithms are search procedures which are known to perform quite well when dealing with large search spaces, such as the ones encountered in object recognition. A set of increasingly complex scenes is used to demonstrate the performance of the genetic search approaches. Our results are promising with exact and near-exact matches being found reliably and quickly.

Keywords: object recognition, affine transformations, genetic algorithms

1. Introduction

The problem of object recognition is a fundamental one in computer vision. During the last two decades, a variety of approaches have been proposed to tackle the problem of object recognition. The most successful approach is in the context of *model-based* object recognition, where the environment is rather constrained and recognition relies upon the existence of a set of predefined set of objects (models) [1]. Given an unknown scene, recognition implies: (i) the identification of a set of features from the unknown scene which approximately match a set of features from a known view of a model object, and (ii) the recovery of the geometric transformation that the model object has undergone (pose recovering). In general, recognition is considerably more difficult because (i) due to occlusions, certain model features will have no corresponding image features and, (ii) due to unknown objects present in the scene, certain image features will have no corresponding model features.

In general, object recognition approaches operate in one of the following two spaces: the *image space* [2] or the *transformation space* [3]. Image space techniques compute the geometric transformation which aligns the model with the scene by first matching a set of model features to a set of scene features. The main problem with this approach is that the number of model-scene feature matches becomes exponential as the number of scene features increases. On the other hand, transformation space techniques search the space of the parameters of the transformation directly, without needing to match model features to scene features first. Although this approach avoids matching model features to scene features, searching the transformation space is also very computationally expensive due to the large number of possible transformations. We elaborate more on these two approaches in Section 2.

In this paper, we propose using Genetic Algorithms (GAs) for recognizing real, planar, objects from two-dimensional intensity images, assuming that the viewpoint is arbitrary. In particular, we consider two different approaches: genetic search in the image space (GA-IS) and genetic search in the transformation space (GA-TS). GAs are search procedures which have shown to perform quite well when the space to be searched is very large [4][5]. Both the image and transformation spaces are large making the use of GAs well suited. GAs operate iteratively on a population of structures, each one of which represents a candidate solution to the problem at hand. Each structure, is modified in a much the same way that populations of individuals evolve under natural selection. Variations among the individuals in the population result in some individuals being more fit than others (i.e., better solutions).

In the past, GAs have been used to solve various problems in computer vision. For example, they have been used for feature selection [6], image segmentation [7], and target recognition [8]. There have also been several attempts to apply GAs in object recognition [9][10]. In [9], GAs were used to solve the object recognition problem which, however, was formulated as a structural matching problem (i.e., graph matching problem). The results reported in [7] are encouraging but are based on artificial objects only. In [10], point patterns were recognized using genetic search in the image space, assuming similarity transformations (i.e., fixed viewpoint). Although that work has similarities with our first approach (GA-IS), here we consider the more general case of affine transformations (i.e., unconstrained viewpoint). Also, our experiments involve real objects whereas the results reported in [10] are based on artificial data only. In a different application, GAs were used for image registration [11]. The main idea in that work was to apply genetic search in a transformation space to register a set of images. This approach has also similarities with our second approach (GS-TS), however, here we try to solve a different problem. Also, we do not apply genetic search in the whole transformation space but we restrict it in a subspace of the transformation space only. To find this subspace, we use a procedure, that we have proposed recently [12][13], to estimate the ranges of values that the parameters of the transformation (affine) assume.

Although there are many problems for which genetic algorithms can find a good solution in reasonable time, there are also problems for which they are inappropriate. These are mainly problems for which it is important to find the exact global optimum. It is well known that genetic algorithms do not perform well in these cases. In the context of object recognition, we expect genetic algorithms to be able to perform at least a rough alignment of the model with the scene. In fact, our experimental results demonstrate that the genetic algorithms find almost exact matches in the case of scenes with little occlusion while they find near-exact matches when scenes with more occlusion are considered. Although near-exact matches might not solve the recognition problem completely, they are useful in the sense that can actually reduce the search space to a limited domain. Then, a local optimization technique can be used for finding an exact match. The preliminary stage of rough alignment may help preventing such methods from reaching a local minimum instead of the global one.

The paper is organized as follows: In Section 2, we review the problem of model-based object recognition and we present some background material. In section 3, we present the genetic object recognition approaches. Specifically, we describe the encoding mechanism, the selection scheme, genetic operators, and fitness function used. Section 4 includes our experiments while our conclusions, limitations, and extensions of the current work are given in section 5.

2. Model-Based Object Recognition

2.1. Affine Transformations

Under the assumption of weak perspective projection (i.e., orthographic projection plus scale), two different views of the same planar object are related through an affine transformation [2]. Specifically, let us assume that an object is characterized by a list of "interest" points $(p_1, p_2, \cdots, p_M)$, which may correspond, for example, to curvature extrema or curvature zero-crossings [14]. Let us now consider two views of the same planar object, V and V', each one taken from a different viewpoint. Also, let us consider two points $p = (x, y)$, $p' = (x', y')$, one from each view, which are in correspondence (i.e., correspond to the same model point). Then, the coordinates of p' can be expressed in terms of the coordinates of p, through an affine transformation:

$$p' = Ap + b \qquad (1)$$

where A is a non-singular 2 x 2 matrix and b is a two-dimensional vector.

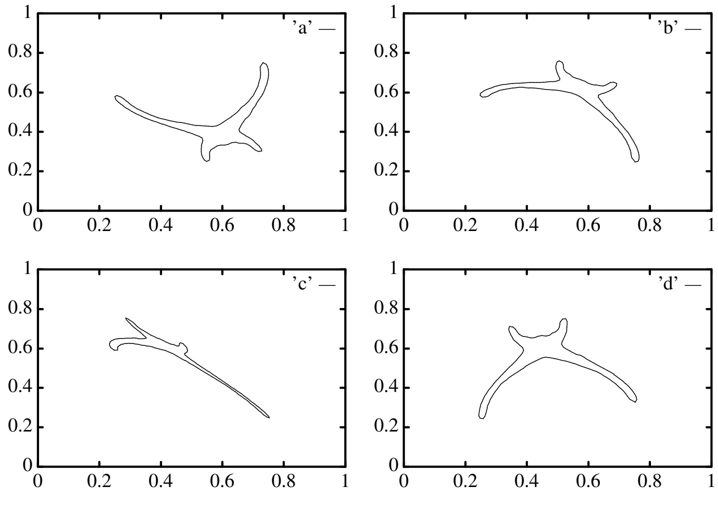

Figure 1. (a) a known view of a planar object (b)-(d) some new, affine transformed, views of the same

object.

Writing (1) in terms of the image coordinates of the points we have:

$$x' = a_{11}x + a_{12}y + b_1 \tag{2}$$

$$y' = a_{21}x + a_{22}y + b_2 \tag{3}$$

Figures 1(b)-(d) show affine transformed views of the planar object shown in Figure 1(a).

2.2. Solution Spaces

Recognition approaches usually operate in one of the following two spaces: the *image space* or the *transformation space*. Image space techniques proceed by first extracting a collection of scene features. Then, a correspondence between these features and a set of model features is hypothesized. The transformation which aligns the model with the scene is determined by this hypothesis. In the case of affine transformations, at least three point correspondences between the model and the scene are required. Since there is usually no *a-priori* knowledge of which model features correspond to which scene features, recognition can be computationally too expensive, even for relatively simple scenes. Assuming M model points and S image points, the maximum number of possible alignments is of the order of $O(M^3S^3)$. Usually, due to errors in the location of the scene points due to noise, more than three point correspondences are required to compute the affine transformation more accurately. In this paper, we consider only three point correspondences. Our first approach to solve the object recognition problem involves using genetic search in the image space (GA-IS) to find the best set of three point correspondences efficiently.

Alternatively, transformation space techniques deal with the space of possible geometric transformations between the model and the scene. The objective is to search the space of all possible transformations in order to find a transformation which would bring a large number of model points into alignment with the scene. In the case of affine transformations, the transformation space is six-dimensional. Again, searching this space is very computationally expensive due to the large number of possible transformations. Our second approach to solve the object recognition problem involves applying genetic search in the transformation space (GA-TS) to find the best transformation efficiently.

Due to the fact that the transformation space is much larger than the image space, genetic search in the transformation space will be significantly slower. Without bounding the parameters of the affine transformation, it will be rather difficult for the GA to find a solution quickly. To speed up genetic search, it is important that we restrict searching in a subspace of the transformation space only. Recently, we have proposed a method, based on Singular Value Decomposition (SVD) [15] and Interval Arithmetic (IA) [16], for estimating the ranges of values that the parameters of affine transformation can assume [12][13]. Here, we use this approach to restrict genetic search in a subspace of the transformation space only. We briefly describe this approach in the next subsection.

2.3. Estimating the parameter ranges

Considering all the "interest" points which comprise V and V', (2) and (3) can be rewritten as follows:

$$\begin{bmatrix} x_1 & y_1 & 1 \\ x_2 & y_2 & 1 \\ \cdots & \cdots & \cdots \\ x_M & y_M & 1 \end{bmatrix} \begin{bmatrix} a_{11} \\ a_{12} \\ b_1 \end{bmatrix} = \begin{bmatrix} x'_1 \\ x'_2 \\ \cdots \\ x'_M \end{bmatrix} \tag{4}$$

$$\begin{bmatrix} x_1 & y_1 & 1 \\ x_2 & y_2 & 1 \\ \cdots & \cdots & \cdots \\ x_M & y_M & 1 \end{bmatrix} \begin{bmatrix} a_{21} \\ a_{22} \\ b_2 \end{bmatrix} = \begin{bmatrix} y'_1 \\ y'_2 \\ \cdots \\ y'_M \end{bmatrix} \tag{5}$$

Using matrix notation, we can rewrite (4) and (5) as $Pc_1 = p_{x'}$ and $Pc_2 = p_{y'}$ correspondingly, where P is the matrix formed by the x- and y-coordinates of the points in V, c_1 and c_2 represent the parameters of the transformation, and $p_{x'}$, $p_{y'}$, are the x- and y-coordinates of V'. Both (4) and (5) are overdetermined and can be solved using a least-squares approach such as SVD [8]. Using SVD, we can factor P as follows:

$$P = UWV^T \tag{6}$$

where both U and V are orthogonal matrices, while W is a diagonal matrix whose elements w_{ii} are always non-negative and are called the singular values of P. The solutions of (4) and (5) are then given by $c_1 = P^+p_{x'}$ and $c_2 = P^+p_{y'}$, where P^+ is the pseudo-inverse of P which is equal to $P^+ = VW^+U^T$, where W^+ is also a diagonal matrix with elements $1/w_{ii}$, if w_{ii} greater than zero (or a very small threshold in practice), and zero otherwise. Specifically, the solutions of (4) and (5) can be written as follows ([15]):

$$c_1 = \sum_{i=1}^{3} \left(\frac{u_i p_{x'}}{w_{ii}}\right) v_i \tag{7}$$

$$c_2 = \sum_{i=1}^{3} \left(\frac{u_i p_{y'}}{w_{ii}}\right) v_i \tag{8}$$

where u_i denotes the i-th column of matrix U and v_i denotes the i-th column of matrix V. Of course, the sum should be restricted for those values of i for which $w_{ii} \neq 0$.

To determine the range of values for c_1 and c_2, we assume that V' has been scaled such that its x- and y-coordinates belong to a specific interval. This can be done, for example, by mapping the image of the unknown view to the unit square. In this way, its x- and y- image coordinates are mapped to the interval [0, 1]. To determine the range of values for c_1 and c_2, we need to consider all possible solutions of (4) and (5), assuming that the p_x and p_y belong to [0,1]. To solve this problem, we use IA [16]. In IA, each variable is represented as an interval of possible values. Given two interval variables $t = [t_1, t_2]$ and $r = [r_1, r_2]$, then the sum and the product of these two interval variables is defined as follows [16]:

$$t + r = [t_1 + r_1, t_2 + r_2] \tag{9}$$

$$t * r = [min(t_1 r_1, t_1 r_2, t_2 r_1, t_2 r_2), \quad max(t_1 r_1, t_1 r_2, t_2 r_1, t_2 r_2)] \tag{10}$$

Applying the interval arithmetic operators to (7) and (8) instead of standard arithmetic operators, we can compute interval solutions for c_1 and c_2 by setting $p_{x'}$=[0,1] and $p_{y'}$=[0,1]. In interval notation, we want to solve the systems $Pc_1 = p^I_{x'}$ and $Pc_2 = p^I_{y'}$, where the superscript I denotes an interval vector. The solutions c^I_1 and c^I_2 should be understood to mean c^I_1=[c_1: $Pc_1 = p_{x'}$, $p_{x'} \in p^I_{x'}$] and c^I_2=[c_2: $Pc_2 = p_{y'}$, $p_{y'} \in p^I_{y}$]. It should be mentioned that since both (7) and (8) involve the same matrix P and $p_{x'}$, $p_{y'}$ assume values in the same interval, the interval solutions c^I_1 and c^I_2 will be the same. Table 1 (first row) shows the range of values computed for c_1 in the case of the object shown in Figure 1(a).

Table 1. The computed ranges of values.

Ranges of values			
	range of a1	range of a2	range of a3
original	[-2.953, 2.953]	[-2.89, 2.89]	[-1.662, 2.662]
preconditioned	[-0.408, 0.408]	[-0.391, 0.391]	[0.0, 1.0]

It can be shown that the width of the ranges depends on the condition of the matrix P and that V can be "preconditioned" to narrow the ranges of values [13]. By pre-conditioning we imply a transformation that can transform V to new view, yielding tighter ranges of values. Table 1 (second row) shows the new, tighter, ranges of values obtained by pre-conditioning V.

3. Methodology

For testing, we used three scenes, Scene1, Scene2, and Scene3, with the object to be recognized being completely visible in Scene1 and increasingly occluded in Scene2 and Scene3 (see Figure 2). Our selection strategy was cross generational. Assuming a population of size N, the offspring double the size of the population and we select the best N individuals from the combined parent-offspring population for further processing [17]. This kind of selection does well with small populations and leads to quick convergence (sometimes prematurely). We also linearly scale fitnesses to try maintain a constant selection pressure.

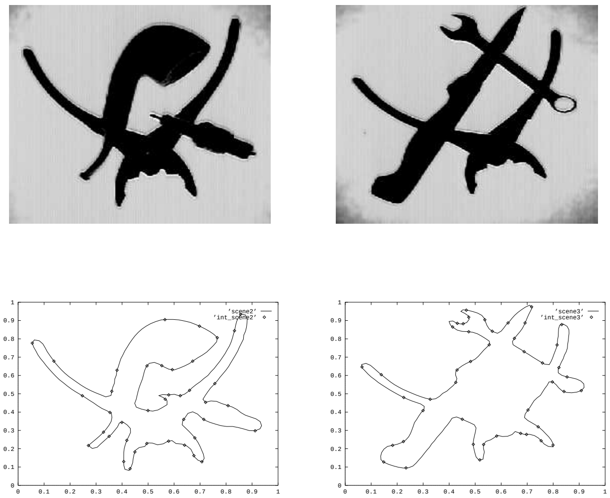

Figure 2. Scenes used in our recognition experiments along with their extracted boundaries and interest points.

3.1. Image Space Encoding

A simple encoding scheme where the identity of the points to be matched made up the alleles in the genotype gave good results on all three scenes. The parameters are provided in the next section. This encoding is shown in Figure 3. Since three pairs of points are needed to compute an affine transformation, the chromosome contains the binary encoded identities of the three pairs of points. The model or object to be recognized was defined by 19 points requiring $\lceil \log 19 \rceil = 5$ bits per point while the scene had between 19 and 45 points and required either 5 or 6 bits per point. We did not check for repeated points and used simple two-point crossover and point mutation.

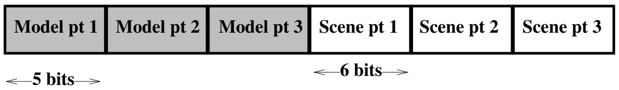

Figure 3. The chromosome contains the binary encoded points to be matched between the model and scene.

3.2. Transformation Space Encoding

A simple binary encoding scheme was also used to represent solutions in the transformation space. Each chromosome contains six fields with each field corresponding to one of the six parameters of the affine transformation. Since we have a methodology to estimate the minimum and maximum value of each parameter, only the range (difference between the maximum and minimum values) needs to be represented. For example, a_{11} assumes values in the interval [–0.408, 0.408] (see second row of Table 1). Thus, its range is $r = 0.408 - (-0.408) = 0.816$. Assuming that up to two decimal points are important in the estimation of the affine transformation, 82 possible values ($\lceil 0.816x100 \rceil + 1$) must be encoded. This means that 7 bits are enough to encode a_{11}'s range. However, 7 bits can be used to represent values from 0 to 127 while we only need to represent values from 0 to 81. As a result, it is possible for the genetic algorithm to find solutions which are not within the desired range (i.e., [0, 81]). To deal with this problem, a simple transformation is used to map values in [0, 127] to values in [0, 81]. Assuming that W is a binary encoded solution corresponding to a_{11}, a_{11}'s actual value can be obtained as follows:

$$a_{11} = MIN(a_{11}) + (82/2^7)) * Decimal(W)$$

where $MIN(a_{11}) = -0.408$ and $Decimal(W)$ is the decimal representation of W. The decoding of the other parameters is performed in a similar way. The constant $(82/2^7)$ is used to map values from [0, 127] to [0, 81]. A simple two-point crossover and point mutation were used again.

3.3. Fitness evaluation

We evaluate fitness of individuals by computing the back-projection error (BE) between the model and scene. Specifically, to evaluate the goodness of the match specified by an individual in the case of the GA-IS approach, we first compute the parameters of the affine transformation which maps the encoded model points to the encoded scene points. This requires the solution of a system of six equations with six unknowns (see (4) and (5)). After the transformation has been computed, we apply it to all the points of the model in order for us to back-project the model onto the scene. Finally, we compute the error, BE, between the back-projected model and the scene. To compute BE, for every model point we find the closest scene point and we compute the distance dj between these two points. Then, the back-projection error is

$$BE = \sum_{i=1}^{M} {d_j}^2 .$$

Since we need to maximize fitness but minimize the error, our fitness function is

$$Fitness = 10000 - BE$$

and changes the minimization problem to a maximization problem for the GA.

To evaluate individuals in the case of the GA-TS approach, we follow exactly the same procedure except that we do not have to solve for the transformation. This is because the transformation is directly encoded in the chromosome and all that is required is to decode it by following the procedure outlined in subsection 3.2.

4. Simulations and results

All genetic algorithm parameters were identical except for the population size and running time. We used a crossover probability of 0.95, a mutation probability of 0.05, and a scaling factor of 1.2. The population sizes were set to 100, 200, and 500 for scenes Scene1, Scene2, and Scene3 respectively. The number of generations needed in the case of the GA-TS approach were twice as many than in the case of the GA-IS approach. In particular, less than 30 generations were required by the GA-IS approach for Scene1, and less than 50 for Scene2 and Scene3. In the case of the GA-TS approach, approximately 100 generations were required for each scenes. This is probably justified by the fact that the transformation space is larger than the image space. However, although the GA-TS approach required more generations to converge to a good solution, this does not mean that it is necessarily slower than the GA-IS approach. The reason is that we do not have to solve for the affine transformation in the case of the GA-TS approach since it is directly encoded in the chromosomes. For each scene, we ran each approach 10 times with different random seeds. Performance plots indicate that the GA very quickly gets close to the correct mapping and then spends most of its time making little progress.

Scene1 was chosen to be exactly the same as our model, (i.e., the goal of the genetic algorithm was to find the identity mapping). Exact mappings (i.e., back-projection error equal to zero) were found using the GA-IS approach in 9 out of 10 trials. Figure 4 (top, left), shows the best and worst solutions found. The solid lines corresponds to the scene and the dashed lines correspond to the back-projected models, using the solutions found. In the case of the best solution, the back-projected model and the scene overly each other and present a single outline. Figure 4 (bottom, left), shows the best fitness (solid line) and average fitness (dashed line) corresponding to the best solution found. As it can be observed, the GA-IS approach found the solution within the first 10 generations or so. In the case of GA-TS approach, almost exact mappings (i.e., small back-projection error but not equal to zero) were found in all cases. Figure 4 (top, right), shows the best and worst solutions found by this approach. In this case, the dashed lines (bak-projected models) do not overly with the solid line (scene) since the back-projection error is non-zero as mentioned above. Figure 4 (bottom, right), shows again the best and average fitnesses for the best solution found by the GA-TS approach. Comparing them with the performance curves for the GA-IS case, it is obvious that the convergence of the GA-TS approach is slower.

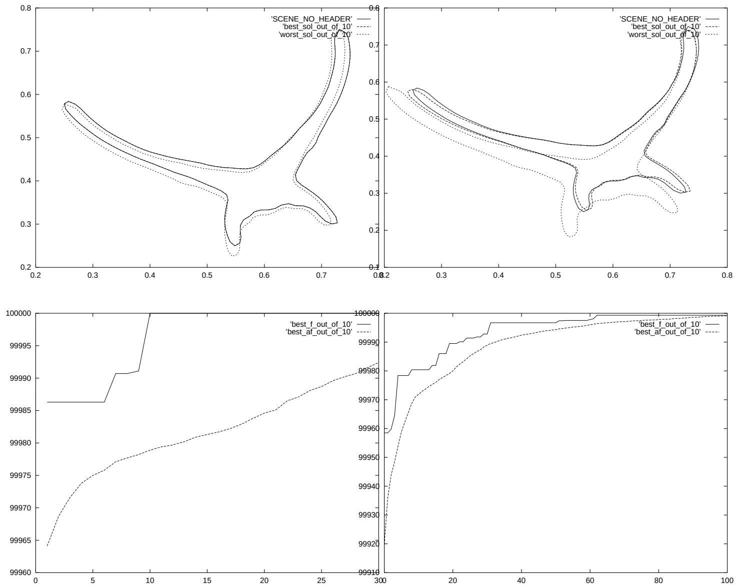

Figure 4. Best and worst solutions found by GA-IS (left) and GA-TS (right) on Scene1 (performance plots are shown on the second row).

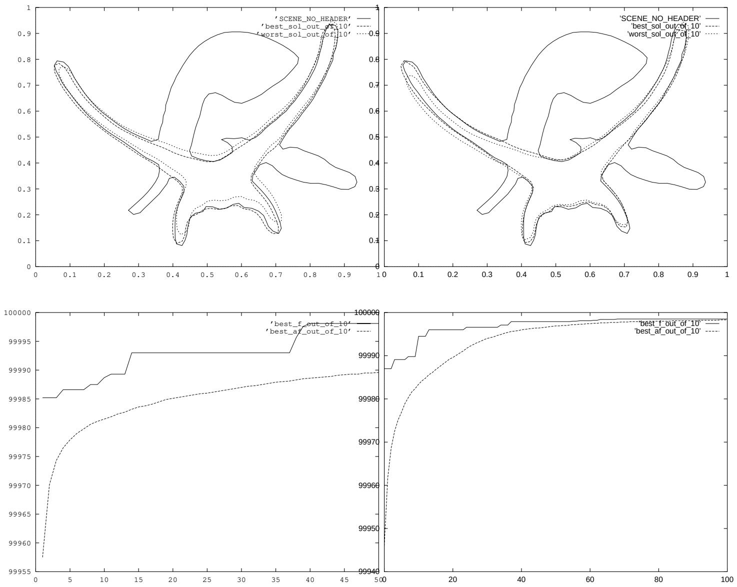

Figure 5. Best and worst solutions found by GA-IS and GA-TS on Scene2 (performance plots are shown on the second row).

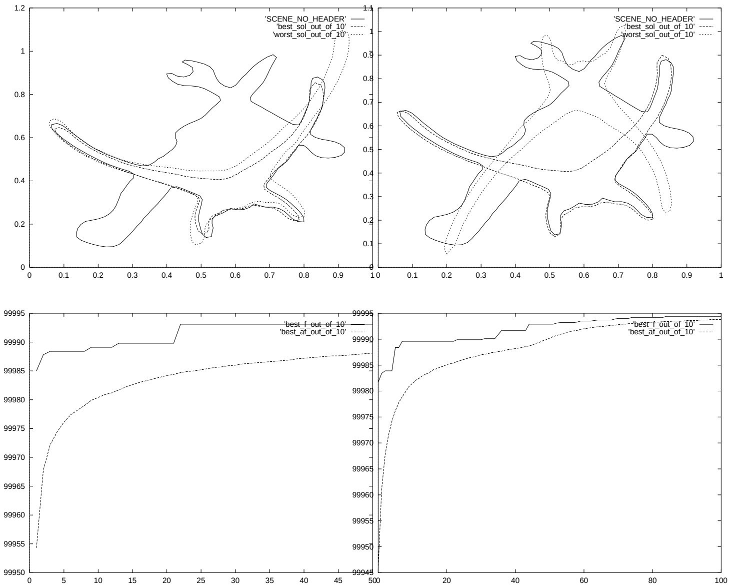

Figure 6. Best and worst solutions found by GA-IS and GA-TS on Scene3 (performance plots are shown on the second row).

Results for Scene2 and Scene3 are shown in Figures 5 and 6. The best and worst solutions found by the two approaches in the case of Scene2 are quite comparable. In the case of Scene3, the GA-IS approach found the correct solution in all 10 trials while the GA-TS approach missed the correct solution once. This case is actually shown in Figure 6 (top, right). It should be mentioned, however, that Scene3 is considerably more difficult than Scene2 since a larger part of the boundary of the object we are trying to recognize is missing. Additional experiments have shown that the wrong solution can be recognized correctly by increasing the population size and the number of generations. Performance plots in both cases illustrate similar results: GA-IS converges faster to a correct solution than GA-TS.

Table 2 provides a summary of our results in the case of the GA-IS approach. The first column specifies the scene. The second and third columns help describe the size of the test problems. The columns list (in order) the number of scene points and the size of the search space. The last two columns indicate GA-IS effort in terms of the number of matches explored and the corresponding fraction in the searched space.

Table 2. Summary of results (GA-IS approach).

Results			
Scene	Scene Points	Number of Matches	$GA - IS_{matches}$
Scene1	19	5,633,766	1800(0.0003)
Scene2	40	57,442,320	47,800(0.0008)
Scene3	45	82,500,660	133,250(0.0016)

In our experiments, M the number of model points is 19, thus the number of possible triplets M_3

$$M_3 = \binom{19}{3} = 969$$

The order of points matters in computing the total number of possible matches between model and scene points and is thus given by the expression

$$Total\ number\ of\ matches = 3!\ x\ M_3\ x\ S_3$$

Referring again to Table 2, the third column lists the total number of possible matches, and the last two columns indicate the average number of matches the GA-IS searched through ($GA - IS_{matches}$), for each encoding. The number within parenthesis calculate *(Total number of matches/*$GA - IS_{matches}$*)* which is the fraction of the space searched by the GA-IS.

Table 3 presents similar results for the case of the GA-TS approach. Referring to Table 1 (second row), the number of values we need to consider in order to represent a_{11}'s range is 82 (see also our discussion in section 3.2). In the case of a_{12}, we need to consider 79 values while in the case of b_1, we need to consider 101 values. The values for a_{21}, a_{22} and b_2 are the same (same interval solutions - see our discussion in section 2.3). This means that the total number of possible transformations is $82^2 x 79^2 x 101^2$=$428,079,701,284$. The last column of Table 3, indicates the number of matches the GA-TS approach searched through. The number in the parenthesis has the same meaning as before.

Table 3. Summary of results (GA-TS approach).

Results		
Scene	Number of Number of Transforms	$GA - TS_{matches}$
Scene1	428,079,701,284	8010 (0.000000018)
Scene2	428,079,701,284	8760(0.00000002)
Scene3	428,079,701,284	8620(0.00000002)

5. Conclusions

In this paper, we considered using genetic algorithms to recognize real, planar, objects from intensity images, assuming that the viewpoint is arbitrary. Two different approaches were considered: genetic search in the image space and genetic search in the transformation space. Our experimental results demonstrate that genetic algorithms are a viable tool for searching these spaces efficiently. One limitation of our current work is that we consider only one model object in our experiments. For future research, we plan to consider more model objects as well as to extend the proposed

-- --

approaches for recognizing 3D objects.

References

[1] R. Chin and C. Dyer, "Model-based recognition in robot vision", *Computing Surveys*, vol. 18, no. 1, pp. 67-108, 1986.

[2] D. Huttenlocher and S. Ullman, "Recognizing solid objects by alignment with an image", *International Journal of Computer Vision*, vol. 5, no. 2, pp. 195-212, 1990.

[3] D. Ballard, "Generalizing the hough transform to detect arbitrary patterns", *Pattern Recognition*, vol. 13, no. 2, pp. 111-122, 1981.

[4] J. Holland, *Adaptation in natural and artificial systems*, The University of Michigan Press, 1975, Ann Arbor.

[5] D. Goldberg, *Genetic algorithm in search, optimization, and machine learning*, Addison-Wesley, 1989, Reading, MA.

[6] G. Roth and M. Levine, "Geometric primitive extraction using a genetic algorithm", *IEEE Transactions of Pattern Analysis and Machine Intelligence*, vol. 16, no. 9, pp. 901-905, 1994.

[7] D. Swets and B. Punch, "Genetic algorithms for object localization in a complex scene", Michigan State University.

[8] A. Katz and P. Thrift, "Generating image filters for target recognition by genetic learning" *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 16, no. 9, 1994.

[9] M. Singh, A. Chatterjee, and S. Chaudhuri, "Matching structural shape descriptions using genetic algorithms", *Pattern Recognition*, vol. 30, no. 9, pp. 1451-1462, 1997.

[10] N. Ansari, M. Chen, and E. Hou, "A genetic algorithm for point pattern matching", in *Dynamic, Genetic, and Chaotic Programming*, Branko Soucek and the IRIS Group (Ed.), pp. 353-371, 1992.

[11] J. Fitzpatrick, J. Grefenstette, and D. Gucht, "Image registration by genetic search", *1984 IEEE Southeastcon*, pp. 460-464.

[12] G. Bebis and M. Georgiopoulos and N. da Vitoria Lobo and M. Shah, "Learning affine transformations of the plane for model-based object recognition", *Proceedings of the 13th International Conference on Pattern Recognition (ICPR-96)*, vol. VI, pp. 60-64, 1996.

[13] G. Bebis and M. Georgiopoulos, M. Shah, and N. da Vitoria Lobo, "Algebraic functions of views for model-based object recognition", *6th International Conference on Computer Vision (ICCV-98)*.

[14] F. Mokhtarian and A. Mackworth, " A theory of multi-scale, curvature-based shape representation for planar curves", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 14, no. 8, pp. 789-805, 1992.

[15] W. Press et. al, *Numerical recipes in C: the art of scientific programming*, Cambridge University Press, 1990.

[16] R. Moore, *Interval analysis*, Prentice-Hall, 1966.

[17] L. Eshelman, The CHC Adaptive Search Algorithm: How to Have Safe Search When Engaging in Non-traditional Genetic Recombination, *Proceedings of the Foundations of Genetic Algorithms Workshop - 1*, Morgan Kauffman, Gregory J. E. Rawlins (ed.), 1990, San Mateo, CA.