Object detection for computer vision using a robust genetic algorithm

Tania Mezzadri Centeno, Heitor Silvério Lopes,
Marcelo Kleber Felisberto, Lúcia Valéria Ramos de Arruda

CPGEI/CEFET-PR, Av. 7 de setembro, 3165,
CEP: 80230-000 Curitiba-PR-Brazil
{mezzadri, hslopes, mkf, arruda}@cpgei.cefetpr.br

Abstract. This work is concerned with the development and implementation of an image pattern recognition approach to support computational vision systems where is necessary to automatically check the presence of specific objects on a scene, and, besides, to describe their position, orientation and scale. The developed methodology involves the use of a genetic algorithm to find target patterns in the image. The proposed approach is fast and presented a robust performance in several test instances including multiobject scenes, with or without partial occlusion.

1 Introduction

Detecting and describing how a specific object appears in an image using traditional matching procedures usually involves hard computational effort, particularly when rotation, translation and scale factors are necessary. In addiction, the complexity of the object recognition problem increases when it is possible to have the target object partially occluded [1].

This work reports the development and implementation of an image matching method, based on a genetic algorithm, aimed at supporting a robust computational vision system. The final objective is to check automatically the presence of specific objects in a scene, describing their position, orientation and scale.

In the proposed approach, image processing techniques are used to extract properties from an object image in order to construct a compact object model representation. Then, a genetic algorithm manages the search for occurrences of the known object model in other images. Next, using the results found by the genetic search, the recognized objects are correctly extracted from the tested images. Thanks to the compact object representation form in our approach, few amount of data are processed and, consequently, less computational effort is spent in the search process.

In this work the identification of specific components in the images is treated as an object verification problem, a particular image analysis case. The goal is to find where and how an object model appears in the input image. According to [1] matching procedures, such as template matching, morphological approaches and analogical meth-

ods, offer feasible solutions for the problem. However, the implementation of a fast search algorithm for this problem is indeed a hard task, since the object can appear in different rotational angles and scales in the image. In this way, this problem can easily fall into an exhaustive search [1].

1.1 Related Work

Fitzpatrick, Grefenstette, and Van Gucht [2][3] faced a similar problem, when they implemented a system for comparison of medical X-ray images to identify dye-coated region in arteries, after the dye injection. The problem, however, was to align both images for comparison by means of images subtraction. The solution was to implement a genetic algorithm to find the best image transformation parameters, in order to align the pre-injection and post-injection images to compare themselves.

According to [4], genetic algorithms are especially appropriate for optimization in large search spaces, where exhaustive search procedures are not feasible. They propose a solution, based on genetic search, for partial image matching problem, applied for medical images.

Bhanu and Peng [5] proposed a method for adaptive image segmentation. They implemented a genetic algorithm to search a set of parameters for image edge-detection. The parameters set were evaluated based on the performance of an object recognition system.

2 Methodology

The first steps in the proposed approach are the definition of the object model representation and the search objective. Next, the genetic algorithm parameters shall be detailed, with special emphasis in the encoding and the computation of the fitness function.

2.1 Object Model Representation

Here we describe the steps of the procedure to encode the image model, representing the pattern, in a compact object representation form, in such a way that a reference matrix and two distances, *dx* and *dy*, fully represents the object model.

a. The image (Fig. 1a) is sliced by n horizontal lines evenly spaced by *dy* pixels, where $dy = number\ of\ image\ lines\ /\ (n+1)$.
b. Similarly as before, the image is sliced by n vertical lines evenly spaced by *dx* pixels, where $dx = number\ of\ image\ columns\ /\ (n+1)$.
c. Crossing lines define points that are named reference points, and are represented by P_{ij}, where $i = 0, 1, \ldots, n-1$, and $j = 0, 1, \ldots, n-1$.
d. The point P_0 at coordinates (x_0, y_0), with $x_0=y_0=(n-1)/2$, is named the central reference point.

e. A function $f(P_{ij})$ assigns to each reference point the mean value of the pixels in the P_{ij} neighborhood, as defined by the delimited region shown in Fig. 1b.
f. All the computed $f(P_{ij})$ values are normalized in the integer range [0..99] and represented as a $n \times n$ matrix, called reference matrix ($\boldsymbol{M}_{ref}$).

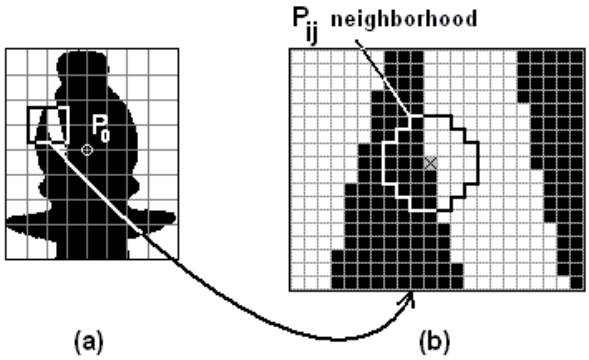


Fig. 1. (a) The image model sliced by seven horizontal and seven vertical lines with the central reference point (P_o); (b) Neighborhood of a generic point P_{ij}.

2.2 Genetic Algorithms

2.2.1 Individual Encoding

In our approach, five parameters are necessary to describe a simple individual: a threshold value t for the input image, a scale factor s, a rotation angle θ, and the pair of coordinates (x_0'; y_0') for the central reference point (P_0') in the input image. Hence, the k-th individual of the population will be represented by the 5-tuple ($x_0'_k$, $y_0'_k$, s_k , θ_k , t_k), whose ranges are shown in the Table 1.

For the tests, detailed later, we shall use images with no more than 2047 lines or columns. Therefore, 11 bits will be enough to represent the central reference point coordinates. For the sake of simplicity, we used the same length for the remaining parameters, t, s and θ. Therefore, an individual will have 44 bits long, leading to a search space $>10^{16}$.

Table 1. Ranges of parameters encoded in an individual k.

Parameter	Range
Po' column	$0 \le x_0'_k \le 2047$
Po' lines	$0 \le y_0'_k \le 2047$
scale factor	$0.5 \le s_k \le 2.0$
rotation angle	$0 \le \theta_k \le 2\pi$ rad
threshold	$0 < t_k < 255$

2.2.2 Objective and Fitness Functions

Firstly, the input image is binarized, based on the threshold value t_k. For the pixels of the image with value less than t_k is assigned the value 0 (black) and, for the remainder pixels, 255 (white).

Based on the parameters encoded in an individual, the coordinates of a generic point (P_{ij}') in the input image is given by Equations 1 and 2, considering translation and rotation [6], respectively:

$$x_i' = x_0' + s.[(x_i - x_0).\cos\theta + (y_i - y_0).\sin\theta] \quad (\mathbf{1})$$

$$y_i' = y_0' + s.[(y_i - y_0).\cos\theta + (x_i - x_0).\sin\theta] \quad (\mathbf{2})$$

where: x_i and y_i are the coordinates of point P_{ij} relative to point P_0 (in the object model image), and the x_i' and y_i' are the coordinates of the point P_{ij}' (projection of point P_{ij} in the input image). For the central reference point P_0', Equations 1 and 2 give P_0' = (x_0'; y_0').

For instance, suppose that the individual represented by vector (23; 311; 6.28; 1, 56) has been generated as a possible solution for the search of the object model in Fig. 1. The reference points would be located as shown in Fig. 2a, for that input image. For better visualizing the result, Fig. 2b shows the object model projection over the input image matched with the proposed solution. Also, in Fig. 2a, it is shown that it is possible some reference points fall off the image limits. Such points are called invalid points, and the result of $f(P_{ij}')$ is represented by an asterisk. The total of invalid points is denoted by n*.

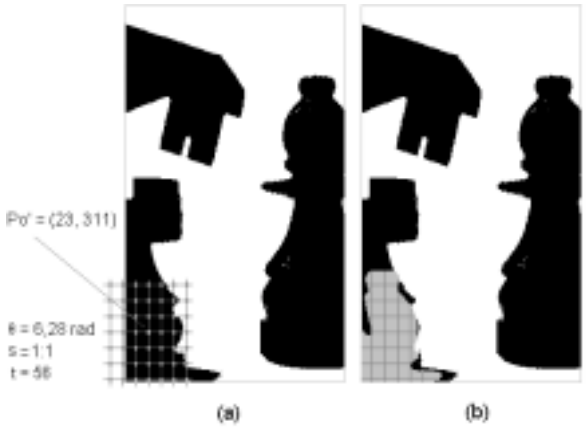


Fig. 2: (a) Reference points plotted in a test image, for individual (23; 311; 6.28; 1, 56); (b) The object model projection in the input image matched with the proposed solution.

Once reference points have been located, a new reference matrix can be generated for the proposed solution, by following steps (e) and (f) of section 2.1. Such matrix, for the *k*-th individual, is denoted by $\boldsymbol{M}_{ref}'$ ($x_0'_k$; $y_0'_k$; s_k ; θ_k ; t_k).

Equation 3 is the objective function that measures the distance between $\boldsymbol{M}_{ref}'$ and $\boldsymbol{M}_{ref}$. It is based on the sum of the quadratic errors and, the small the distance $S_{SQE}(k)$, the better is a given solution *k*.

By default, the fitness function of a genetic algorithm deals with a maximization problem. Since it is searched for a given solution *k* ($x_0'_k$; $y_0'_k$; s_k ; θ_k ; t_k) that minimizes $S_{SQE}(k)$, a new fitness function is defined in Equation 5:

$$fit(k) = \frac{S_{SQE-MAX} - S_{SQE}.(k)}{S_{SQE-MAX}} \tag{5}$$

where:

$$S_{SQE-MAX} = (99 \cdot (n \times n - n^*))^2 \tag{6}$$

Here, $n \times n$ is the $\boldsymbol{M}_{ref}$ dimensions and n^* is the number of invalid points of reference.

Since $S_{SQE\text{-}MAX} \geq S_{SQE}(k)$ for any feasible solution *k*, the fitness function values will be defined within the range [0...1].

A restriction to the maximum number of invalid points was incorporated in the fitness function definition to limit the maximum number of n^* occurrences. Therefore, the fitness function is redefined, as follows, considering the index "*WR*" as the fitness value with restriction:

$$\begin{cases} if \quad \text{n}^* > \dfrac{n \times n}{2} \Rightarrow fit_{WR}(k) = fit(k) \\ \\ Otherwise \quad \Rightarrow fit_{WR}(k) = 0 \end{cases} \tag{7}$$

2.2.3 The Genetic Search

The genetic search starts with the random generation of the initial population of *z* individuals: ($x_0'_1$; $y_0'_1$; s_1 ; θ_1; t_1), ($x_0'_2$; $y_0'_2$; s_2 ; θ_2; t_2), … , ($x_0'_z$; $y_0'_z$; s_z ; θ_z; t_z). Each individual of the population is evaluated by the fitness function, and the probability of each individual to be selected for reproduction increases proportionally to its fitness value. An appropriated selection method [7] is used to select candidates for crossover and mutation. Such genetic operators will generate new individuals to form a new population. Some population generating strategies include elitism that means to copy some of the fittest individuals of the current population to the next one. Basically these same procedures are used to generate the following populations until some stop criterion is met. Usually a maximum number of generations or a satisfactory fitness value reached is used as stop criterion.

3 Implementation

The system was implemented in C++ object-oriented programming language on Microsoft Windows 2000 platform. For the image processing routines we used the Dilabiem 6.11 package [8], and for the GA implementation, the GAlib genetic algorithm package [9]. Fig. 3 shows a block diagram that illustrates the information flow through the system components, which are described as follows

a) The parameters updating block is used to modify running parameters of the GA.
b) The model construction block applies operators to the object model image in order to construct the object model representation (M_{ref}, *dx, dy*), as explained before.
c) The genetic search block uses a GA to find a promising region of the input image that supposedly contains the object model. Such solution is represented by a vector as described in Section 2.2.1.
d) The object image extraction block makes a decision based on the fitness value of the current solution. For a fitness value less than a fixed fitness threshold (t_{FIT}), the solution is just discarded and the search stops. Otherwise, the object is extracted from the input image and saved as a new image denominated *object image i* (*i*=1… *j*). Besides, the result of the image subtraction (*input image – object image i*) is feed-backed to the genetic search block, for a new search.

The loop between the blocks genetic search and object image extraction allows the system to find further copies of the same object in the input image.

5 Experiments and Results

Several experiments were done using images of chess pieces. An image of a single object was used as model. Fig. 4 shows the object model image used in the experiments and its reference matrix (***M****ref*), generated by the model construction block (Fig. 3). Other images, where the same object appears in different orientations and positions was tested, as well as multiobject scenes and partial occlusion occurrences.

During the GA run, the number of solutions generated and evaluated is $z \times g = 100 \times 1000 = 10^5$. The search space, considering a typical 512 × 512 pixels image, 3600 possible values for the rotation angle (0.0°, 0.1°, 0.2°,…, 359.9°), and 101 scale factors (0.500, 0.515, 0.530, …, 2.000), is larger than 9.53×10^{10}. Consequently, the genetic algorithm can find an acceptable solution testing less than 0.0002 % of that huge search space.

Fig. 5 shows one of the images used in the experiments, where the target object appears in three different locations and positions in the same image. Table 2 shows the search results for the experiments using the image of Fig. 5 as input. Each table column shows the object extracted from the original input image, followed by the parameters found in the genetic search, and the fitness value for the current solution. These results show that all occurrences of the target object in Fig. 5 were found. Note that the partially occluded object (*object image #1*), in the right side of the image (Fig. 5), was also detected and correctly extracted from the image.

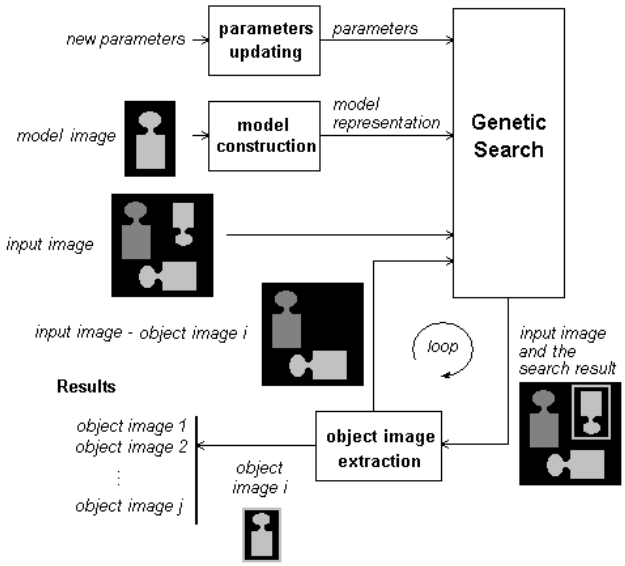
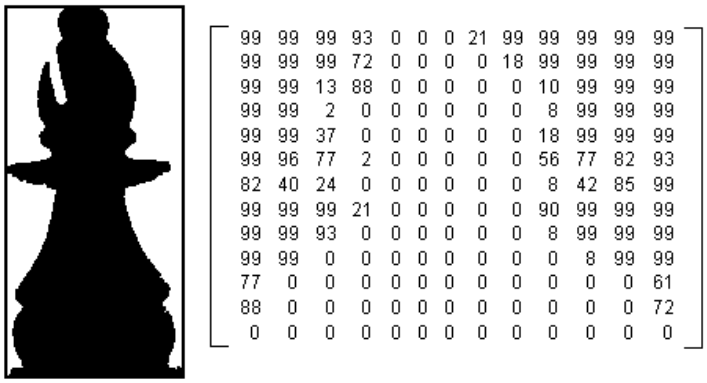


Fig. 3. Block diagram of the proposed system showing its components and the corresponding information flow.



99	99	99	93	0	0	0	21	99	99	99	99	99
99	99	99	72	0	0	0	0	18	99	99	99	99
99	99	13	88	0	0	0	0	0	10	99	99	99
99	99	2	0	0	0	0	0	0	8	99	99	99
99	99	37	0	0	0	0	0	0	18	99	99	99
99	96	77	2	0	0	0	0	0	56	77	82	93
82	40	24	0	0	0	0	0	0	8	42	85	99
99	99	99	21	0	0	0	0	0	90	99	99	99
99	99	93	0	0	0	0	0	0	8	99	99	99
99	99	0	0	0	0	0	0	0	0	8	99	99
77	0	0	0	0	0	0	0	0	0	0	0	61
88	0	0	0	0	0	0	0	0	0	0	0	72
0	0	0	0	0	0	0	0	0	0	0	0	0

(a) (b)

Fig. 4. (a) Object model image used in the experiments. (b) Corresponding reference matrix.

Fig. 6 shows another test image used, where the target object appears in a multiobject scene. Fig. 7 shows the result of the object model matching with the solution proposed by the system.

Table 3 shows the results for the experiments using as input the image shown in Fig. 6. Note that the Table 3 shows the two next solutions that would be found by the system if the fitness threshold value t_{FIT} would decrease from 0.85 to 0.80.

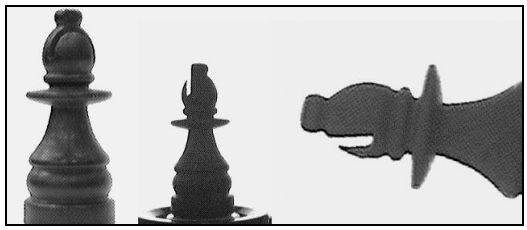
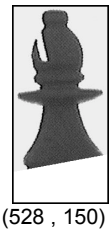
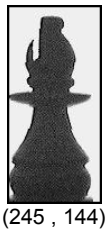


Fig. 5. Images used in the experiments.

Table 2. Results obtained using the image of the Fig. 5 as an input.

object image #1	object image #2	object image #3
		
P_0' = (528 , 150) s = 1.523926 θ = 1.386719 rad t = 140 fitness = 0.902464	P_0' = (80, 113) s = 0.934570 θ = 6.273985 rad t = 160 fitness = 0.894589	P_0' = (245 , 144) s = 0.747803 θ = 0.033748 rad t = 118 fitness = 0.907113

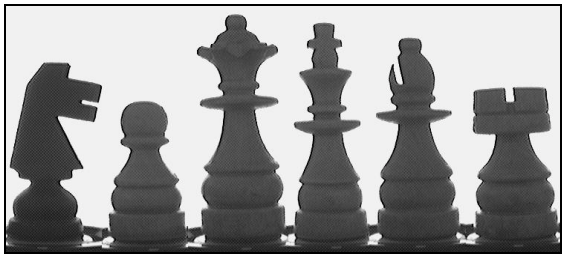


Fig. 6. Multiobject image used in the experiments.

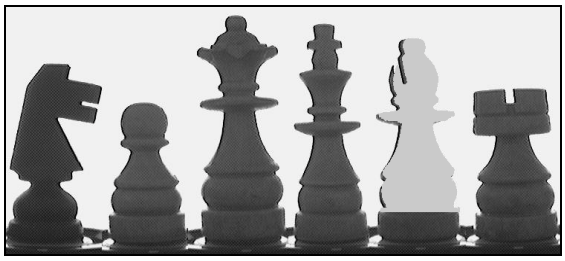
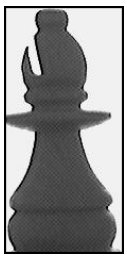
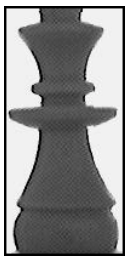
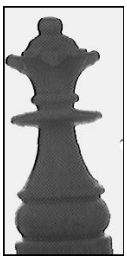


Fig. 7. Matching of the first solution proposed by the GA, using as input the image in Fig. 6.

Table 3. Results obtained using as input the image shown in Fig. 7.

object image #1	Discarded solution #1	discarded solution #2
		
P_0' = (555 , 152) s = 0.983545 θ = 0.000000 rad t = 128 fitness = 0.881313	P_0' = (433 , 169) s = 0.983545 θ = 0.012272 rad t = 125 fitness = 0.832775	P_0' = (321 , 138) s = 1.222607 θ = 6.258645 rad t = 238 fitness = 0.844708

6 Conclusions

This work proposed an image pattern recognition approach based on a genetic algorithm. The implemented system is the core of an upcoming computer vision system. This approach is useful in many cases where it is necessary to check the presence of specific objects in a scene, and, further, to describe their position, orientation and scale.

During the reported experiments, the system displayed a good performance for all test sets, demonstrating its robustness. However, it was observed that, for most cases, the solutions presented by the system were not the optima, but something very close to the optimum value (see, for instance, Fig. 7). The developed system is computationally efficient, obtaining good solutions in few seconds, for all test images.

Despite the good results, future work will focus accuracy, including the implementation of some local search strategy in order to fine-tune results from the genetic search, leading to even more accurate results.

Acknowledgements

This work has been partially supported by Agência Nacional do Petróleo (ANP) and Financiadora de Estudos e Projetos (FINEP) - ANP/MCT (PRH10-CEFET-PR).

References

1. Jain R, Kasturi R and Schunck BG (1995) *Machine Vision*. McGraw-Hill, New York
2. Fitzpatrick JM, Grefenstette JJ and Van-Gucht D (1984) Image registration by genetic search. In: *Proc Southeastcon*, vol. 84, pp. 460–464
3. Grefenstette JJ and Fitzpatrick JM (1985) Genetic search with approximate function evaluations. In: *Proc Int Conf on Genetic Algorithms and their Applications*, pp. 112–120
4. Simunic KS and Loncaric S (1998) A genetic search-based partial image matching. In: *Proc 2nd IEEE Int Conf on Intelligent Processing Systems*, pp. 119-122
5. Bhanu B and Peng J (2000) Adaptive integrated image segmentation and object recognition. *IEEE T Syst Man Cy C* 30(4):427–441
6. Gonzalez RC and Wintz P (1987) *Digital Image Processing*, Addison-Wesley, Boston
7. Bäck T and Hoffmeister F (1991) Extended selection mechanisms in genetic algorithms. In: *Proc 4th Int Conf on Genetic Algorithms*, pp. 92–99
8. Osowsky J (2004) DILabiem *Image Processing Package*. Technical Report vs. 6.11, CPGEI/CEFET-PR
9. Wall M (2003) *GAlib A C++ Library of Genetic Algorithm Components* vs. 2.4.5. http://lancet.mit.edu/ga/.