Particle Swarm Optimization for Object Recognition in Computer Vision*

Hugo A. Perlin, Heitor S. Lopes, and Tânia Mezzadri Centeno

Bioinformatics Laboratory, Federal University of Technology Paraná (UTFPR),
Av. 7 de setembro, 3165 80230-901, Curitiba (PR), Brazil
haperlin@gmail.com, hslopes@pesquisador.cnpq.br, mezzadri@utfpr.edu.br

Abstract. Particle Swarm Optimization (PSO) is an evolutionary computation technique frequently used for optimization tasks. This work aims at applying PSO for recognizing specific patterns in complex images. Experiments were done with gray level and color images, with and without noise. PSO was able to find predefined reference images, submitted to translation, rotation, scaling, occlusion, noise and change in the viewpoint in the landscape image. Several experiments were done to evaluate the performance of PSO. Results show that the proposed method is robust and very promising for real-world applications.

1 Introduction

Automatic pattern recognition in images is an important problem of computer vision, frequently found in industry, security, engineering and other areas. The main objective is to find specific objects (patterns or reference images) in a complex scene subject to degradation of quality. Many image processing methods have been used for this problem, mainly based on mathematical morphology or template matching [1]. Most image processing techniques are computationally expensive and too specific for certain types of images. Also, the lack of robustness limits their application to noisy images and when the object to be searched is occluded or randomly displaced/rotated in the scene. Overall, to achieve satisfactory performance for real-world applications, exhaustive search methods are inadequate and, then, heuristic methods can play an important role [2,3,4].

Particle Swarm Optimization (PSO) [5] is an evolutionary computation method, and has been successfully used for hard numerical and combinatorial optimization problems [6,7]. When compared with other heuristic optimization methods, PSO is easier to implement and tune parameters. PSO has been used very sparsely for image processing problems, mainly for image registration. This problem consists in aligning geometrically two images in such a way to optimize

* This work was partially supported by the Brazilian National Research Council – CNPq, under research grants nos. 309262/2007-0 to H.S. Lopes, and 477922/06-6 to T.M. Centeno. Authors also acknowledge the financial aid from CAPES as a scholarship to H.A. Perlin.

N.T. Nguyen et al. (Eds.): IEA/AIE 2008, LNAI 5027, pp. 11–21, 2008.

(maximizing) a similarity measure. PSO was shown to be robust and efficient for this kind of optimization [8,9].

This work presents the implementation of PSO method for detecting objects in images in the context of computer vision. Given a pattern or reference image of an object, it is to be found anywhere in a target image (complex scene). Once found the object, its location is precisely determined in the target image, including planar coordinates, scale and rotation angle.

2 The Object Recognition Problem

Given two images, the first one containing a reference object (pattern), and the another as a target landscape where the object is supposed to be found, finding the pattern in the target is defined as finding the planar coordinates, rotation angle and scale factor. A possible solution is represented as 4-tuple: (x, y, s, θ), where x and y represent the planar coordinates of the center of the reference image relative to the landscape image, s is the scale factor, and θ is the rotation angle, relative to the coordinates system. Since the object can appear anywhere in the landscape, the set of all possible combinations is very large. Therefore, the problem can be stated as an optimization problem, so as to find appropriate values for (x, y, s, θ) that maximizes the similarity between the object and landscape images (see below).

In this work, the search space is limited by constraining the range of possible values for the variables of the problem, as follows: column ($0 \leq x < n$), row ($0 \leq y < m$), scale ($0.5 \leq s \leq 2.0$), and rotation ($-\pi \leq \theta \leq \pi$).

To evaluate possible solutions for the problem, it is necessary to define a measure of similarity between the reference and the landscape images. Some measures of similarity between images were proposed in the literature, such as mutual information and sum of the square of differences between pixels [8,9]. Since mutual information is computationally expensive and the other measure may give high values for similarity even when images are not correlated, we used in this work the absolute sum of differences between pixels.

A function that computes the difference between pixels of the reference image (RI) and the landscape image (LI) is called here "$Error$". Using such function a given solution can be evaluated by using the following equations:

$$Eval_{Sol} = \frac{(Error_{Max} - Error_{Sol})}{Error_{Max}} \tag{1}$$

$$Error_{Max} = 2^{nbits} * ((m * n) - Pinv) \tag{2}$$

$$Error_{Sol} = \sum_{i=0}^{n-1} \sum_{j=0}^{m-1} |RI(i,j) - LI(I,J)| \tag{3}$$

where: n and m are the dimensions (width and height) of the reference image; $nbits$ is the number of bits used to represent gray levels; $Pinv$ are pixels that

belongs to the reference image, but doesn't belongs to the landscape image (they appear when the position of the reference image is near to the borders of landscape image); $Error_{Sol}$ is the sum of the difference of intensity between pixels of reference and landscape images, for the given solution.

The indices (I, J) of the pixels of the landscape image are obtained by the equations 4 and 5, where $ddX = j - \frac{Width_{RI}}{2}$ and $ddX = i - \frac{Heigth_{RI}}{2}$, and the values of x, y, s, θ belong to the candidate solution under evaluation.

$$I = y + s * (ddX * sin(-\theta) + ddY * cos(\theta)); \tag{4}$$

$$J = x + s * (ddX * cos(-\theta) + ddY * sin(\theta)); \tag{5}$$

The evaluation function (Equation 1) tends to 1, its maximum value, when the value of Equation 3 tends to 0. Therefore, an optimization method (in this case, PSO) can be used to maximize the value of the evaluation function, consequently minimizing the error between both reference and landscape images(Equation 3).

The previous definitions is for gray level images, although a similar principle can be used for color images. In this case, it is necessary to compute the matching error for each independent channel of the triplet RGB, as shown in the following equations:

$$Eval_{Sol} = \frac{(3 * Error_{Max} - (ErrorChannel_{Sol}))}{Error_{Max}} \tag{6}$$

$$ErrorChannel_{Sol} = \sum_{ch=1}^{3} \sum_{i=0}^{n-1} \sum_{j=0}^{m-1} |RI(i, j, ch) - LI(I, J)| \tag{7}$$

An important difference for the evaluation of color images is the range of values of Equation 6, from 0 to 3. However, this feature does not lead to significant differences in the recognition of objects in images. Considering that color images have three independent color channels, represented by the variable *ch* in the equation 7, the search for objects can be simplified if the landscape image is previously searched for large color discrepancies. This procedure can decrease the searchable size of the landscape image, thus significantly decreasing the computational effort.

3 Particle Swarm Optimization (PSO)

Particle Swarm Optimization (PSO) belongs to the group of heuristic methods known as swarm intelligence. PSO is an evolutionary computation method, like genetic algorithms, genetic programming, evolution strategies and ant colony optimization. PSO is inspired in the behavior of social agents, and was invented by Eberhart and Kennedy in 1995 [5]. The basic idea behind PSO is the simulation of a simplified social system, based on the collective behavior observed in bird flocking, bee swarming, and fish schooling, for instance. The individuals in these groups continuously adjust their position in the space while moving, so

as to keep an average distance between neighbors. The behavior of a given individual affects the group and vice-versa. From the computational point of view, the swarm is composed by particles, which represent possible solutions for the problem. Particles "fly" over the hypersurface of the solution space, searching for the optimal solution.

At the beginning, the population of particles is randomly initialized. That is, their position in the search space, as well as their velocity (in all dimensions) are set randomly. Using a random number generator with uniform distribution, such initialization assures that any point in the search space can be reached. Each particle has a limited memory, and it is able to store the coordinates of its current position in the search space, as well as the position where it found the best solution to date (*pbest*), and the position of the best solution found by its neighbors (*lbest*) or by the whole swarm (*gbest*), depending on the implementation. *pbest* represents the knowledge acquired by the own particle during its navigation in the search space, also known as cognitive component. *gbest* or *lbest* represents the knowledge acquired by the group, also known as social component.

At each time step, the movement of particles is influenced by both its cognitive as well as its social components, as follows. Each particle has a velocity, which is modified according by the weighted influence of *pbest* and *gbest*. The larger is the difference between the current position of the particle to *pbest*, the more it is influenced to go towards it. The same occours for *gbest*. Equation (8) shows how the velocity of the i-th particle is updated in the next time step $(t+1)$, according to its current position x_i^t at time t.

$$Vel_i^{t+1} = w * Vel_i^t + c_1 * r_1 * (pbest_i^t - x_i^t) + c_2 * r_2 * (gbest_i^t - x_i^t) \quad (8)$$

where: w is the inertia moment, c_1 and c_2 are user-defined acceleration constants, r_1 and r_2 are uniformly distributed random numbers, and x_i represents the current solution.

Knowing the velocity of a given particle, it is possible to update its position in the search space at the next time step. This is done by Equation (9).

$$x_i^{t+1} = x_i^t + Vel_i^{t+1} \quad (9)$$

The acceleration constants, c_1 and c_2, have a direct influence on the size of the step in the search space. Therefore, it is important to set such constants to appropriate values, aiming at an efficient exploration of the search space. A high value for c_1, local search is encouraged and the swarm tends to be clustered into small isolated groups, frequently with only one particle. On the other hand, with a high value for c_2, the swarm tends to be clustered in a local maximum, thus stagnating the search. As any other evolutionary computation method, the choice of optimal values for the control parameters is a difficult task and, usually, is problem-dependent.

The following steps summarize the standard PSO algorithm:

a) Initialize the swarm of particles with random values for each component of the solution vector (valid solution);
b) Evaluate the quality of the solution represented by each particle;

c) If the fitness of the current solution is better than *pbest* then update it with the current solution. If the current solution if better than *gbest* then update it with the current solution;
d) Compute the new velocity of particles according to Equation (8);
e) Compute the new position of particles (new solution) in the search space according to Equation (9);
f) If a stopping criterion is met, then stop. Otherwise, go to step (b). The stopping criterion can be a maximum number of iterations or a solution of satisfactory quality.

An additional step that can be used to improve performance of PSO is the use of some diversification method within the search process. When the swarm converges to a given region of the search space, very few improvement can be achieved. This is due to the lack of diversity of solutions in the swarm and can be observed when *gbest* stagnates for a number of iterations. In such situation, the easiest way to give chance of a further improvement of the PSO is to extinguish the swarm and restart PSO, but keeping the former *gbest*. This procedure is known as explosion, and was demonstrated to be of great utility for complex problems [6], allowing PSO to achieve better solutions, when compared with a PSO without explosions.

In fact, the PSO heuristics is easy to implement and does not require many computational resources for running. The most expensive part is always the evaluation of a candidate solution (particle).

4 Computational Experiments and Results

The standard PSO was implemented following the steps summarized in the section 3. A population of 40 particles, with random initialization, was used. Running parameters were set as the default values suggested in the literature: $c_1 = c_2 = 2$ and $w = 0.5$ The stopping criterion was set as reaching 800 iterations. Additionally, during the search, if premature convergence is detected, an explosion is performed [6]. In this work, premature convergence was defined as 20 iterations without improvement in *gbest*.

4.1 Experiment #1

The objective of this experiment is to investigate the sensitivity of the fitness function to changes in translation (x, y), scale (s) and rotation (θ). Translation for both x and y were changed, independently, in the range of $[-5.0..5.0]$, in steps of 0.1 pixel. Scale was changed in the range [0.5..1.5], in steps of 0.001. Rotation was changed in the range of $[-0.0872..0.0871]$ in steps of 0.00001 radian. This is equivalent to -5 to $+5$ degrees. Several landscapes and reference images (grayscale and color) were used in this experiment. Results, in general, were equivalent. A representative result of this experiments is shown in Fig. 1, using Fig. 4(a) as landscape and 4(c) as reference.

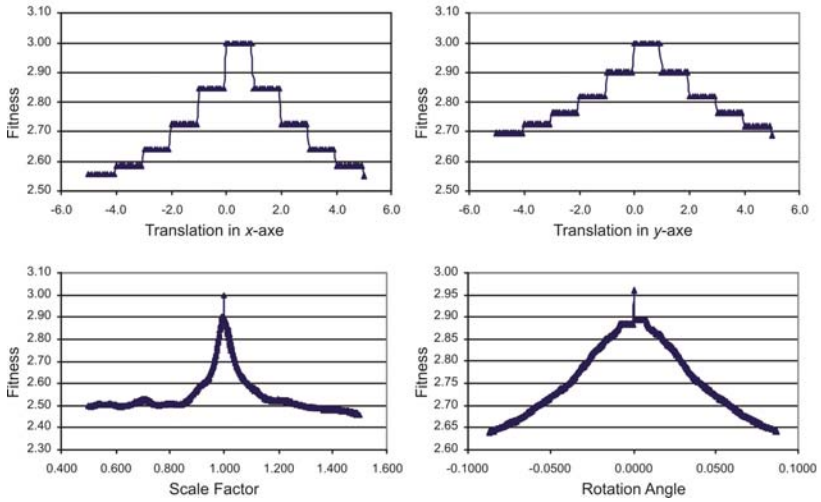


Fig. 1. Sensitivity analysis of the fitness function. Top: translation in axes x and y. Botton: scale and rotation.

4.2 Experiment #2

In this experiment the PSO was tested using a black-and-white reference image, shown in Fig. 2(a), having 69 x 148 pixels (from the work of [2]). Fig. 2(b) and Fig. 2(c) show the landscape images used, with the reference object translated and rotated, but with no change in scale.

(a) (b) (c)

Fig. 2. (a) Reference image. (b) Landscape image with the pattern translated, rotated and degraded with noise. (c) Same landscape image of Fig. 2(b), but with more noise.

Gaussian noise was also added to the analysis image, but with a larger standard deviation (2.0). This experiment enabled to evaluate how PSO degrades its performance as noise increases.

To evaluate in what extent the proposed method is able to find the reference object, we added Gaussian noise to the landscape image. The Gaussian noise added had zero mean and standard deviation 0.5 and 2.0, for the landscape images of Fig. 2(b) and Fig. 2(c), respectively. The size of both landscape images was 208 x 150 pixels. In both cases, PSO found a good solution, shown in Tables 1 and 2. PSO easily found those solutions, converging with only 86 and 94

Table 1. Solution found by PSO for reference image of Fig. 2(a) and landscape image of Fig. 2(b)

Par.	Ref.	PSO	Error
y	84	84	0 %
x	106	107	0.94 %
s	1	1.006251	0.63 %
θ	-1.57	-1.565450	-0.29 %

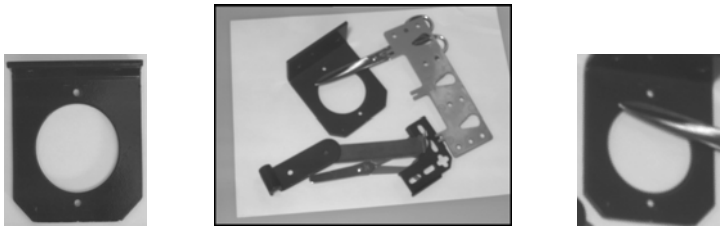
Table 2. Solution found by PSO for reference image of Fig. 2(a) and landscape image of Fig. 2(c)

Par.	Ref.	PSO	Error
y	84	84	0 %
x	106	107	0.94 %
s	1	0.986482	-1.35 %
θ	-1.57	-1.571803	0.11 %

iterations, respectively, achieving a fitness of 0.735727 and 0.612881. In these tables, and in the following, "Par" refers to the elements of the 4-tuple of the image; "Ref" are the reference values; "PSO" refers to the values found by PSO; and "Error" is the percent of deviation of the solution found by PSO, regarding the reference values.

4.3 Experiment #3

To test the robustness of PSO in a more realistic situation, we used a gray level figure with several objects. The reference object, shown in Fig. 3(a) (238 x 269 pixels) is supposed to be found in the scene of Fig. 3(b) (496 x 347 pixels). It should be noted that the object is now translated, rotated, reduced and with a different viewpoint (in perspective). Besides, the reference object is partially occluded by another object (scissor) in the landscape. All these elements configures a hard task for an automatic object recognition system, since it is close to a real-world situation. PSO was able to find a satisfactory solution, shown in Fig. 3(c) (238 x 269 pixels), clipped from the original landscape image. The result parameters found by PSO was 133 and 209, to the x and y coordinates respectively, 0.533792153 for the scale factor and 0.69526328 to the rotation angle.

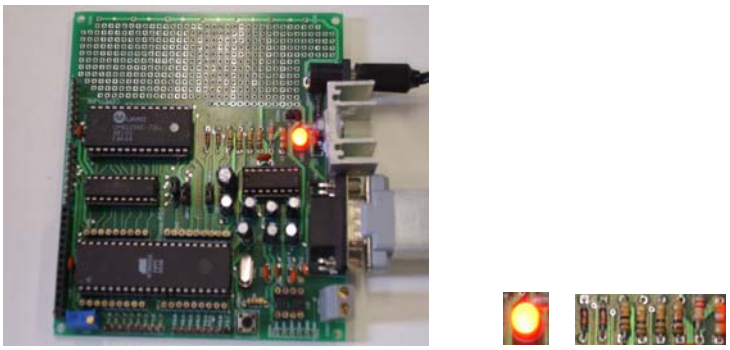


(a) (b) (c)

Fig. 3. (a) Reference image. (b) Landscape image. (c) Clip of the solution found by PSO.

4.4 Experiment #4

Besides gray scale images, we tested the proposed method with color images quantized with 24 bits. First, the reference image of Fig. 4(b) (76 x 83 pixels)



(a) (b) (c)

Fig. 4. (a) Color landscape image of a printed circuit board. (b) Reference image (bright led). (c) Reference image for experiment #1 (resistors).

Table 3. Comparison of the performance of PSO for gray scale and color versions of images of Fig. 4(a) and Fig. 4(b)

Type of Image	Avg. Fitness	Avg. Iter.	Avg. Expl.
Gray level	2.659252754	225.6	3.6
Color	2.911080048	191.5	3.4

and landscape image of Fig. 4(a) (800 x 600 pixels) were transformed to gray scale (using 8 bits) and submitted to PSO. Next, both color images were submitted to PSO in order to compare the effect of the decrement of the quantization level in the efficacy of PSO. Results are shown in Table 3 represent the average of 10 independent runs. The average fitness found for color images was larger than the corresponding for gray scale images. This suggests that, for this set of images, it is easier to find the object in a color image. In fact, it was intentionally selected the object with most discrepancy in the landscape (round, red and bright). However, when the image is converted to gray scale, there is no such discrepancy. Even so, PSO succeed to find the object, but with a larger effort, demonstrated by the average number of iterations necessary. The average number of explosions of the swarm was also larger for the gray scale image than for the color one. This is compatible with the larger difficulty imposed by the first case.

4.5 Experiment #5

In this experiment we used a large color image (1024 x 768 pixels) as landscape – see Fig. 5(a), and a small reference image (118 x 138 pixels) – see Fig. 5(b). The face of the reference figure has many details similar to other faces in the figure. Therefore, this experiment evaluated the ability of the proposed method to find an object in a complex scene with rich color details.

In this experiment, color Gaussian noise was added to the landscape image, in a percentage ranging from 5% to 75%. The objective of this test is to discover

(a) (b)

Fig. 5. (a) Landscape figure of a complex scene. (b) Reference figure.

the limit of efficiency of the PSO. Table 4 shows the average results for 10 independent runs. As expected, the efficiency of PSO decreases as noise increases. A statistical analysis of data shows that this decrement is polynomial. In the same way, as a consequence of the increasing difficulty imposed by noise in the image, the average number of iterations increases also polynomially. The average number of explosions (restarts) is directly correlated to the number of iterations.

5 Discussion and Conclusions

In the experiment #1 it can be observed a difference in sensitivity for axes x and y. The larger sensitivity to translations in axis x is due to the specific features of the reference image 4(c), where elements are vertically positioned. The changes in the fitness function due to translation is in steps because pixels are discrete, although the PSO uses a resolution of tenths of a pixel. In this experiment it was also observed that changes in scale is much more critical than in translation or rotation. This fact suggests the possibility of using some kind of local search just to fine-tune the scale of a solution found by PSO.

Table 4. Results for different levels of color Gaussian noise in the landscape Fig. 5(a)

Noise	Avg. Fitness	Max. Fitness	Avg. Iter.	Avg. Expl.
5 %	2.70863165	2.85806349	208.6	3.6
10 %	2.59230251	2.74718357	267.4	6.1
20 %	2.53714739	2.55977623	198.0	4.6
30 %	2.23639087	2.32247749	248.3	5.1
50 %	2.09965423	2,15037207	229.6	5.4
75 %	1.93363241	1,97322669	418.2	15.2

Experiment #2 showed that PSO is robust, even in the presence of strong noise, when the reference image has a well-defined threshold with the background, typical of black-and-white images. Similarly, for color images, experiment #5 also showed the robustness of PSO, even for a complex scene with many details. The difficulty of the task is directly proportional to the amount of noise.

In the experiment #3 PSO showed its usefulness in a situation very close to real-world applications, specially in the industrial environment. PSO succeeded well in the experiment with a highly unfavorable conjunction of factors: displacement, rotation, reduction, occlusion and different viewpoint of the reference image.

Experiment #4 showed that the proposed method has more difficulty with images with less quantization levels. This suggests that it is easier to find patterns in color images because there is more information in the three channels than in a single gray level channel.

Overall, the problem of automatic object recognition in images is constantly found in industry, robotics, medicine, engineering, computer science and other areas. Therefore, efficient methods for dealing with this problem are always welcome. The proposed method, based on PSO, was shown to be a good alternative for the problem, and we believe it is specially suited for industrial applications, even in adverse situations.

To avoid parameter adjustments without prior knowledge of the behavior of the algorithm for each specific problem, future work will focus on a self-adaptive version of the PSO, inspired in a previous work with genetic algorithms [10]. Also, another evaluation functions will be tested so as to decrease computational effort and improve robustness.

Finally, we believe that the proposed method is promising, specially concerning the robustness and simplicity. Results suggest that this method can be used in a computer vision system for real-world object detection in images.

References

1. Jain, R., Kasturi, R., Schunck, B.G.: Machine Vision. McGraw-Hill, New York (1995)
2. Felizberto, M.K., Centeno, T.M., Lopes, H.S.: Object detection for computer vision using a robust genetic algorithm. In: Rothlauf, F., Branke, J., Cagnoni, S., Corne, D.W., Drechsler, R., Jin, Y., Machado, P., Marchiori, E., Romero, J., Smith, G.D., Squillero, G. (eds.) EvoWorkshops 2005. LNCS, vol. 3449, pp. 284–293. Springer, Heidelberg (2005)
3. Felizberto, M.K., Lopes, H.S., Centeno, T.M., Arruda, L.V.R.: An object detection and recognition system for weld bead extraction from digital radiographs. Comput. Vis. Image Unders. 102, 238–249 (2006)
4. Guerra, C., Pascucci, V.: Find line segments with tabu search. IEICE Trans. Inf. Syst. E84-D(12), 1739–1744 (2001)
5. Eberhart, R.C., Kennedy, J.: Swarm Intelligence. Morgan Kaufmann, San Francisco (2001)

6. Hembecker, F., Lopes, H.S., Godoy Jr., W.: Particle swarm optimization for the multidimensional knapsack problem. In: Min, G., Di Martino, B., Yang, L.T., Guo, M., Rünger, G. (eds.) ISPA Workshops 2006. LNCS, vol. 4331, pp. 358–365. Springer, Heidelberg (2006)
7. Lopes, H.S., Coelho, L.S.: Particle swarm optimization with fast local search for the blind traveling salesman problem. In: Proc. 5th Int. Conf. on Hybrid Intelligent Systems, pp. 245–250 (2005)
8. Talbi, H., Batouche, M.: Hybrid particle swam with differential evolution for multimodal image registration. In: Proc. IEEE Int. Conf. on Industrial Technology (ICIT), pp. 1568–1572 (2004)
9. Wachowiak, M.P., Smolíková, R., Zurada, J.M., Elmaghraby, E.: An approach to multimodal biomedical image registration utilizing particle swarm optimization. IEEE Trans. Evolut. Comput. 8(3), 289–301 (2004)
10. Maruo, M.H., Lopes, H.S., Delgado, M.R.B.S.: Self-Adapting Evolutionary Parameters: Encoding Aspects for Combinatorial Optimization Problems. In: Raidl, G.R., Gottlieb, J. (eds.) EvoCOP 2005. LNCS, vol. 3448, pp. 154–165. Springer, Heidelberg (2005)