Journal of Circuits, Systems, and Computers

TEMPLATE MATCHING IN DIGITAL IMAGES USING A COMPACT GENETIC ALGORITHM WITH ELITISM AND MUTATION*

RAFAEL R. DA SILVA†
CARLOS R. ERIG LIMA§
HEITOR S. LOPES‡

Bioinformatics Laboratory/CPGEI,
Federal University of Technology – Paraná,
Av. 7 de setembro 3165, 80230-901 Curitiba (PR), Brazil
†*rafael.rsi@gmail.com,* §*erig@utfpr.edu.br,* ‡*hslopes@utfpr.edu.br*

Received (30 april 2009)
Revised (30 april 2009)
Accepted (30 april 2009)

The emCGA is a new extension of the Compact Genetic Algorithm (CGA) that includes elitism and a mutation operator. These improvements do not increase significantly the computational cost or the memory consumption and, on the other hand, increases the overall performance in comparison with other similar works. The emCGA is applied to the problem of object recognition in digital images. The objective is to find a reference image (template) in a landscape image, subject to distortions and degradation in quality. Two models for dealing with the images are proposed, both based on the intensity of light. Several experiments were done with reference and landscape images, under different situations. The emCGA was compared with an exhaustive search algorithm and another CGA proposed in the literature. The emCGA was found to be more efficient for this problem, when compared with the other algorithms. We also compared the two proposed models for the object. One of them is more suitable for images with rich details, and the other for images with low illumination level. Both models seems to perform equally in the presence of distortions. Overall, results suggested the efficiency of emCGA for template matching in images and encourages future developments.

Keywords: Compact Genetic Algorithm, Template Matching, Image processing.

1. Introduction

A Genetic Algorithm (GA) is an efficient tool for search and optimization problems, and has been applied to many computational and engineering problems. However, in some applications, the computational cost of the GA can be too high, demanding a prohibitive execution time or even excessive hardware resources. When a GA is

*This work was partially supported by the Brazilian National Research Council (CNPq) under research grant no. 309262/2007-0 to H.S.Lopes.

implemented in software, usually the computational resources demanded are less important. However, for high-performance applications, using a GA implemented in hardware (usually, reconfigurable logic devices[5]), the amount of memory required to store the population can be an important limitation of the implementation. Recall that a GA evolves a population, not a single point, thus requiring memory space to store such information during generations. Another important issue that is raised are the techniques used to evolve this population. These techniques were developed for running in sequential processors, restricting its application in parallel architectures. A feasible alternative is the use of a genetic algorithm with smaller computational complexity, which can run in less powerful systems. The Compact Genetic Algorithm (CGA)[8] can achieve the same level of quality of an ordinary SGA (Simple Genetic Algorithm) with uniform crossover, but using much less memory to store the population. This is possible because the CGA works with a probability vector instead of the whole population.

Another feature of the CGA is the use of techniques to evolve the probability vector, imitating the behavior of a SGA. Due to the simplicity of these techniques and the small memory requirements, some works proposed software and hardware implementations of the CGA, showing good results with significant resources reduction for its construction. This is an important issue because the present work will be the base for a future implementation in reconfigurable hardware. This is a current technological trend for the development of high-performance parallel systems in many areas, such as mobile robotics[3], bioinformatics[12] and image processing[14].

Also, another motivation for this work is that CGA is relatively new, when compared with the traditional GA, and few studies about its performance and applications are available in the recent literature. Therefore, this is an interesting subject to be studied further.

This work presents an application of the emCGA[17], a new extension of a CGA. This approach uses elitism and introduces a new mutation operator. Differently from other mutation operators formerly introduced for CGA, this one is applied to new individual generation, thus imitating the crossover operator of a SGA. This operator does not increase significantly the computational cost or the memory consumption and, on the other hand, increases the overall performance, when compared with similar works. The emCGA is applied to the object recognition problem. Two models are introduced, for color and grayscale images, and several experiments are done to evaluate its performance.

2. The Compact Genetic Algorithm

The CGA is an Estimation of Distribution Algorithm (EDA)[11,19], first proposed by Harik et al.[8], that generates descendants using a statistical population model, instead of the traditional recombination operators and mutation. As result, the population in CGA is reduced to a single vector of probabilities that occupies only $L.log_2(N)$ bits of memory, in comparison with $L.N$ for a SGA (where L is the

chromosome length and N the size of the population). The CGA can imitate the behavior of a SGA with uniform crossover using a reduced amount of memory. The importance of the size of the population in GA performance has been focused in many recent works[7,9,16]. In general, a large population size results in better quality of the solution, but it increases the computation cost and memory use. Due the importance of this parameter, several works aimed at improving the representation of the population, especially for hardware-based applications.

The genetic drift is a term used by the population genetics area that explains the tendency of changes in the frequency of an allele through random samplings in a population. In other words, even with a selective pressure, the frequency of the alleles in the population varies due to stochastic errors associated to the genetic operators. When the stochastic variation in the frequency of the alleles overcomes the selection variation, then a drift stall takes place. In this case, the population will converge to a non optimal solution[15,18]. Consequently, for an optimal or quasi-optimal convergence of the CGA population, the appropriate selective pressure is quite important. Concisely, the occurrence of the drift stall reflects the lack of appropriate selective pressure. This one can be increased in CGA, by increasing the tournament size (of the selection method) or by using elitism.

Since CGA was introduced by Harik et al.[8], several extensions were proposed aiming to improve its performance. Their main focus are the introduction of techniques to allow the CGA to overcome the performance reduction in problems with high-order building blocks (BBs). The extensions of CGA that proposed some elitist technique were those that showed the best balance between demand of resources and performance. Elitism allows to increase the selective pressure and, consequently, to reduce the genetic drift[2].

Despite of the competitive performance of CGA with SGA for low-order BBs, it does not achieves the same performance when high-order BBs are present in the problem. In this case, with the compact representation of the population in CGA, the information regarding high-order relationships among the chromosome bits does not survive throughout generations, differently that what happens in a SGA. For real-world problems, these disturbances generated by the uniform crossover should be overcome, since many problems presents local optima (i.e. multimodal) and interdependent genes (high-order BBs). However, increasing the selective pressure in a CGA tends to decrease these noisy effects, because it increases the probability of high-order BBs to survive throughout generations [1,7]. In other words, the increase of the selective pressure can accomplish the function of the population memory. Some works proposed modifications in the basic CGA to improve performance by increasing the selective pressure, such as: neCGA[1], mCGA[6] and, more recently, emCGA[17]. Among them, the emCGA showed the best tradeoff between solution quality (fitness value) and convergence speed (number of evaluations).

2.1. *The emCGA*

The previously mentioned works (neCGA[1] and mCGA[6]) perform better than the original CGA. Both approaches use elitism, but mCGA also employs a mutation operator. Another CGA that also includes a mutation operator is MBBCGA[20], but it is different from the mCGA mutation operator. In this work, e use a new mutation operator, aimed at improving even more the quality of solutions but, also, keeping a reasonable convergence speed. The previously mentioned works that use mutation operators do not focus on population diversity control, but on local search. In the first work the mutation is applied to the elite individual of the current generation to generate another individual. After a tournament over these individuals, the best one will be the elite for the next generation. In the second work, mutation is applied to the first individual to substitute the second generation of the conventional CGA.

The new mutation operator used here allows a more efficient control of the selective pressure, adjusting the population diversity as the consequence of the manipulation of the probability vector. Comparing the proposed mutation operator with that of mCGA, the new operator decreases the number of tournaments per generation and, consequently, the total number of fitness evaluations per generation. The consequence is a significant improvement in the convergence speed of the algorithm, as pointed out in a recent study[17]. The new CGA resulting from the use of the proposed mutation is named emCGA (elitism with mutation CGA). Basically, the proposed mutation operator changes the random generation phase, by changing the chromosome just-generated with the probability vector.

3. The emCGA for the Object Recognition Problem

The object recognition problem in images is frequently found in real-world and academic applications. Sometimes, this problem is also referred as image registration problem. It consists in finding a given reference image (or template) somewhere in a (much larger) landscape (or input) image. A simple example would be to find a known human face in a crowd of people. The larger he match between the template and a region of the landscape image, the higher the probability it includes the object of interest. In general, the recognition of objects using traditional image processing algorithms is computationally expensive. Particularly, this effort increases significantly when it is present translation, rotation, scale variation or partial obstruction of the reference image in the landscape image[10]. Therefore, the implementation of fast search algorithms for this problem is of great interest. In many cases, applications using these algorithms should be executed in real time (such as in real-time face recognition), thus requiring fast algorithms, instead of exhaustive search algorithms. Algorithms based on metaheuristics (such as those from the evolutionary computation area) can offer reasonable performance and quality of solutions [4,13]. This fact motivated the use of emCGA to the object recognition problem in real-world images. In this section, a technique of digital processing of images is explored to extrapolate significant properties of the image of a reference object, and to create

two computational models. Then, the search of the reference image in the landscape image is accomplished by emCGA, despite its displacement in position and rotation or obstruction.

Images can be represented in several formats, for instance, bitmap. Although the bitmap format is widely known and complete, not all information in the format may be useful for a given application (especially for identifying the image). Therefore, different representations of the images are defined in this work, and two models for approaching the problem were devised, the Light Intensity (LINT) model and the Binary (BIN) model. These models will be detailed in the next sections.

3.1. *The Object Models*

The Light Intensity (LINT) model is based on the intensity of the light in the three channels of the image of the object to be detected, Red, Green and Blue (RGB). These channels are quantisized in 8 bits and were weighted with 0.30, 0.59 and 0.11, respectively, so as to obtain the effective illuminance of a pixel when converting color images to grayscale. The conversion of all pixel of the image yields a generic matrix M_{LINT}, where each element is a value of 256 levels, corresponding to the light intensity of the pixel in the same image position. By definition, when the value of the element is 0 means that the light intensity is 0%, and 255 is 100%.

The Binary (BIN) model is also based on the intensity of light, as in the LINT model. However, this information is binary, that is, either there is light or not, depending on a given threshold. This is accomplished by means of the binarization of the M_{LINT} matrix, resulting in another generic matrix, named M_{BIN}. The binarization process is done for each pixel of the image: if the intensity value is larger than a threshold, the corresponding cell of the matrix is set to 1, otherwise to 0.

3.2. *Fitness Function*

The fitness function of the emCGA measures the similarity between two images. The object is defined by a reference image. The landscape images is previously converted to the desired object model (section 3.1). Therefore, it is possible to reduce the computational cost and the memory demands in the fitness function execution. The fitness function proposed by Perlin et al.[13] computes the percentage of similarity ($S(k)$) between two images, the landscape (L) and the reference (R) image transformed by the set of parameters k. Such parameters include, at least, the planar translation and rotation, and, optionally, scaling. The function shown in equation 2 computes, in fact, the absolute error between the reference and landscape images when the first is properly overlapped over the latter. It is used to compute the final fitness function shown in equation 1.

$$S(k) = 100\%.\frac{E_{max} - E(k)}{E_{max}} \tag{1}$$

$$E(k) = \sum_{i=1}^{n} \sum_{j=1}^{m} \| L_{LINT}(i,j) - \mathfrak{T}(R_{LINT}, k, i, j) \| \quad (2)$$

where: L_{LINT} and R_{LINT} are the LINT matrix of the landscape (L) and reference (R) images, respectively; $\mathfrak{T}$ is a function that transforms an image according to the vector of parameters k and returns the (i, j) element of the transformed image; E_{max} is the maximum absolute error that is computed according to the model used, LINT or BIN (see below); n and m are number of lines and columns of reference image matrix, respectively.

This fitness function returns a value in the range 0..100% representing the similarity between the reference and the landscape image. The higher this value, the higher the probability of finding the object in the landscape image, according to the vector of parameters k. Since all elements of the R matrix is used for computing similarity of the object with the landscape, the objective is to find a vector k that yields the higher possible similarity measured by the fitness function. The above equations 1 and 2 can be used for both models, LINT and BIN, provided the corresponding matrices are computed according to the model.

Objects studied in this work are three-dimensional. Consequently, they can be found displaced horizontally (x axis), vertically (y axis) and in depth (z axis), as well as rotated in any of the three possible planes (angles θ, α and β). Displacement in the z axis corresponds to changes in the scaling of the image, and rotations in the x, z or y, z planes are considered distortions. In this work we considered only translations and rotations in the (x, y) plane. The distortions, usually present in real-world images, are indirectly treated by our approach, since they will cause an increase mismatching of reference and landscape images.

The $\mathfrak{T}$ transform applies translation and rotation operations in (x, y) plane to a given point (i, j) and returns the corresponding element of the reference matrix, defined by coordinates (x', y'), according to equations 3 and 4.

$$x' = x + (i. \cos\theta + j. \sin\theta) \quad (3)$$

$$y' = y + (j. \cos\theta - i. \sin\theta) \quad (4)$$

The position (x', y') is the transformed position of (i, j), according to this parameters vector. However, since this transformation uses trigonometrical functions that return a real number, the returned element of the matrix I is the value of the closest integer of this position, by rounding up x' and y'.

The translation position is a position in the landscape image and it is also the central position of the possible object location. When a position is evaluated close to the border, some transformed positions (x', y') may become invalid. In this case the error in these points will be the maximum. This procedure allows the identification of an object partially clipped close to the borders.

A chromosome in the emCGA is then defined by the parameter vector k and coded according to the following ranges: $0 \leq x \leq c$, $0 \leq y \leq Llin$ and $0 \leq \theta < 360$, where $Lcol$ and $Llin$ are the number of columns and lines of the landscape image.

Displacements are measured in pixels (steps of 1) and angle in degrees (steps of 0.7). Consequently, the overall length of the chromosome for the LINT model is $L = log_2(Lcol) + log_2(Llin) + 9$.

For the BIN model, besides the transformations previously mentioned for the LINT model, it is necessary to establish the binarization threshold t. Therefore, for the BIN model, the vector of parameter for the $\mathfrak{T}$ transformation also includes t. For the BIN model, each element of the matrix will be either 0 or 1, and so, the maximum absolute error possible becomes $E_{max} = n.m$.

3.3. *emCGA Parameters*

The emCGA uses two parameters that need to be previously defined: the population size and the mutation rate. The search of an object in an image is a complex problem, since it may be strongly multimodal, with many local maxima randomly distributed. Besides, it is important to note that the genes in the chromosome, that is, the parameters being optimized, are strongly interconnected. Consequently, to properly cope with such epistasy, the mutation rate recommended is approximately $1/L$, and the size of the population should be large enough. See [17] for a detailed discussion about these subjects.

The landscape images used in our experiments were 1024 x 768 bits large. In this case, parameters x and y will need 10 bits for encoding, plus 9 bits for the rotation angle. Consequently, the recommended mutation rate is approximately 3.4%. After preliminary experiments, the size of the population that yielded a reasonable tradeoff between performance and computational cost was 16384 individuals. Evolutionary computation algorithms, as is the case of emCGA, have their stochastic nature implemented through a pseudo-random number generator. Thus, it is also necessary to define a seed for this generator. For each independent run, a different random seed was chosen.

4. Experiments and Results

Two groups of experiments were done to evaluate the emCGA for the object recognition task in images. The first group is aimed to compare the proposed emCGA with another similar CGA. The objective of the second group is to compare both emCGA models themselves, LINT and BIN.

Our experiments use images from digital pictures. For the sake of having a gold standard, that is, the optimal solution, for a given problem, we devised an Exhaustive Search Algorithm (ESA). ESA evaluates the same fitness function (equation 1) for all possible combinations of the parameters vector k.

4.1. *Comparison of two CGA approaches*

In this experiment, the object to be found in the input image is a chess bishop piece. The reference image was used to generate both the LINT and the BIN models,

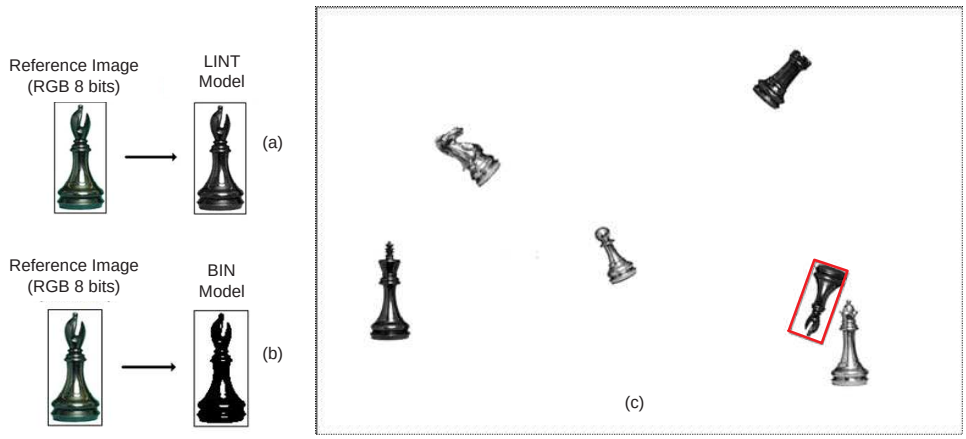

Fig. 1. Chess bishop piece search. a) Reference image and corresponding LINT and BIN models. b) Landscape image and the object recognized.

Algorithm	**Object model**	**x**	**y**	θ	**t**	**S**	**Eval.**
emCGA	LINT	795	209	200.08	-	97.29	8.81E+04
ESA	LINT	795	209	200.08	-	97.29	4.03E+08
emCGA	BIN	795	209	200.08	146	97.83	8.12E+04
ESA	BIN	795	209	200.08	140	97.83	1.03E+11
mCGA	-	737	511	153.58	-	78.68	4.17E+05

Table 1. Results for the comparison of two CGA approaches and exhaustive search algorithm.

according to figure 1a. This model is used by the emCGA to find the object in the landscape image. The object found is presented in figure 1b inside a rectangle to stand it out and facilitate its visualization. This procedure will be also used in the remaining experiments in this work. Notice that the bishop in the input image is translated and rotated relative to the reference image. Both models, LINT and BIN succeeded to find exactly the same result, in this case, the optimal result, as pointed out by the ESA in table 1. Values for the parameters vectors are shown in each experiment as well as the fitness value (S) and the number of fitness evaluations (Eval.) needed to converge the respective algorithm. The comparison of the emCGA models was done with another CGA, namely the mCGA proposed by Gallager et al.[6].

The former experiment was a simple problem. This is promptly realized considering the results obtained by the methods. A harder problem is treated now, where the object to be recognized is a human face within a digital picture in which there are several other people. This is a difficult problem for a computer-based vision

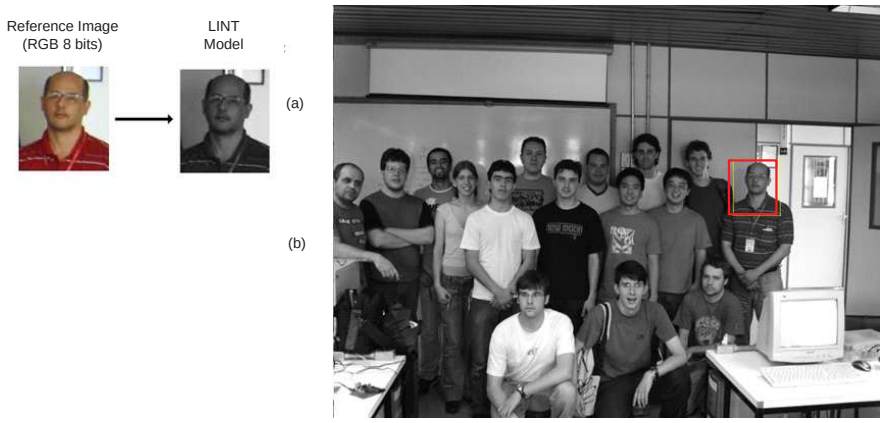

Fig. 2. Real-world faces search. a) Reference image and the associated model. b) Landscape image with the recognized face.

Algorithm	**Object model**	**x**	**y**	θ	**t**	**S**	**Eval.**
emCGA	LINT	792	428	0	-	100	1.08E+05
ESA	LINT	792	428	0	-	100	4.03E+08
mCGA	-	185	293	3.52	-	84.88	3.84E+05

Table 2. Results for the comparison of two object models and exhaustive search algorithm.

system, since there are many local optima. The face used is shown in figure 2a and the landscape image with the corresponding recognized face are shown in figure 2b. The reference image is cut out of the input image, and this image did not undergo any distortion. The corresponding numerical results are presented in table 2.

In the last experiment of this group, we added real-world factors that makes harder the object recognition problem. In this experiment, the object to be recognized is a chess knight piece (figure 3a) and there are two landscape images. In the first landscape image, shown in figure 3b, the object is translated and rotated in the image plane (x, y). The main distortion in this image that makes the problem somewhat difficult is due to a different illumination angle, shedding a bright spot to the object. In the second landscape image (figure 3c), the object is also rotated in both (z, y) and (x, z) axes and mirrored, using the same illumination scheme. These rotations impose a more challenging task than the previous one. Besides these distortions, in the two input images other similar pieces are added, increasing the difficulty of the problem by injecting more local maxima. The numerical results of this experiment are shown in table 3.

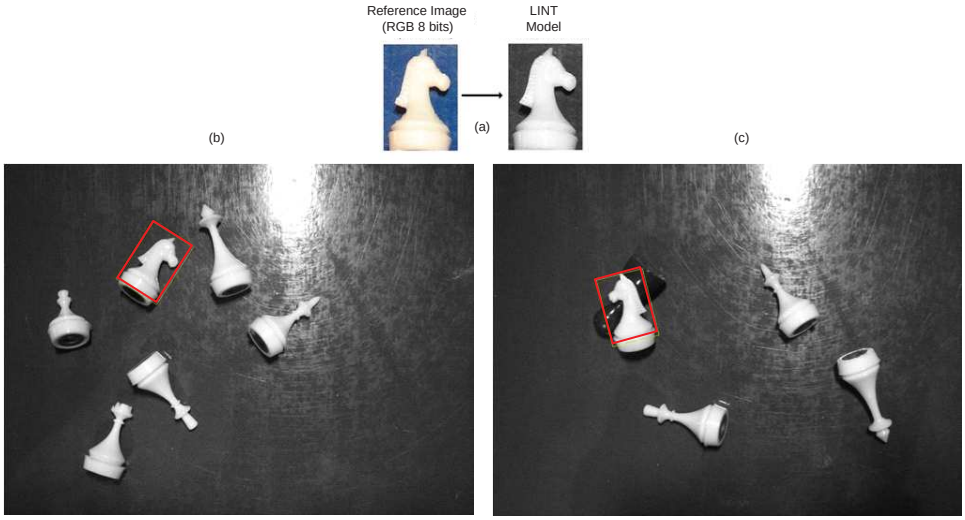

Fig. 3. Chess knight piece search. a) Reference image and the associated LINT model. b) and c) Two different landscape images with the recognized object.

Algorithm	**Object model**	**x**	**y**	θ	**t**	**S**	**Eval.**
emCGA(a)	LINT	327	522	32.41	-	92.86	9.58E+04
ESA(a)	LINT	327	522	32.41	-	92.86	4.03E+08
mCGA(a)	-	452	501	225.44	-	80.63	3.59E+05
emCGA(b)	LINT	292	455	345.91	-	85.04	8.07E+04
ESA(b)	LINT	292	455	345.91	-	85.04	4.03E+08
mCGA(b)	-	292	455	345.91	-	85.04	3.84E+05

Table 3. Results for the comparison of two object models and exhaustive search algorithm.

4.2. *Comparison of the two object models*

The second set of experiments aims at unveiling possible differences between the two proposed models, LINT and BIN. The first experiment uses a domino game piece as reference image, shown in figure 4a for the LINT model, and in figure 4b for the BIN model. Since the target objet to be recognized does not present any distortion in the landscape image, both models obtained similar results, see figure 4c for the LINT model, and figure 4d for the BIN model, respectively. Table 4 shows the numerical results for this experiment.

The next experiment uses basically the same landscape images, but they the are with significant illumination attenuation, both models. The reference images are the same as in figure 4a and 4b. Low level of illumination decreases the average difference between pixels, and makes the identification of the object more difficult.

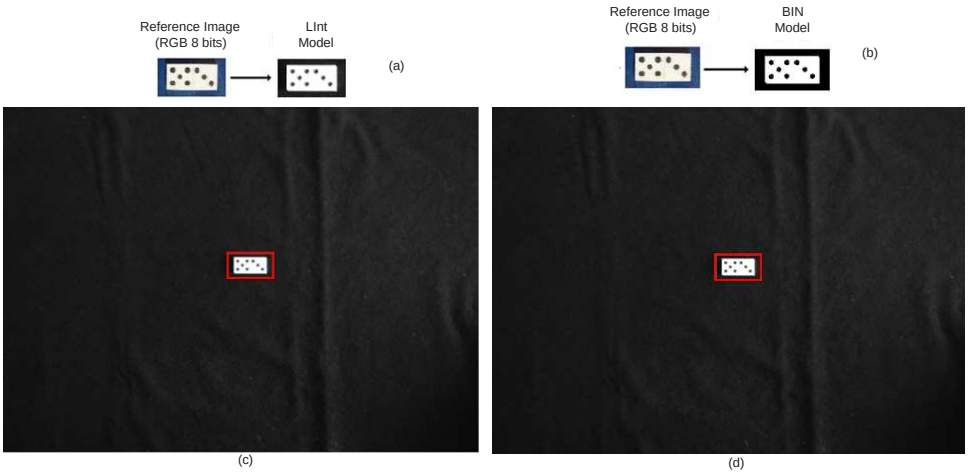

Fig. 4. Domino piece search. a) Reference image and LINT model. b)Reference image and BIN model. c)and d) Landscape images with the object recognized.

Algorithm	**Object model**	**x**	**y**	θ	**t**	**S**	**Eval.**
emCGA	LINT	533	412	0	-	100	8.03E+04
ESA	LINT	533	412	0	-	100	4.03E+08
emCGA	BIN	533	412	360	128	100	1.04E+05
ESA	BIN	533	412	360	128	100	1.03E+11

Table 4. Results for the comparison of two object models and exhaustive search algorithm.

Figures 5a and 5b show the object recognized using the LINT and the BIN models, respectively. In these figures, the object is, actually, positioned within the red box drawn to facilitate visualization. The object was detached and shown in a new box. The corresponding numerical results are in table 5. Notice that the values of fitness are all low, due to the effect of illumination attenuation in the image. For the LINT model, emCGA and ESA found the same object, but rotated around 180 degrees, thus giving a slight difference in fitness.

Finally, using the reference images of figure 4, and the input image of figure 6, the object to be recognized is among several other quite similar pieces, characterizing a difficult problem for a computer-based vision system, with many local optima. Numerical results for this experiments are in table 6.

5. Discussion and Conclusions

The object recognition problem for computational vision easily falls into an exhaustive search. It is expected that the use of a metaheuristic search algorithm, such

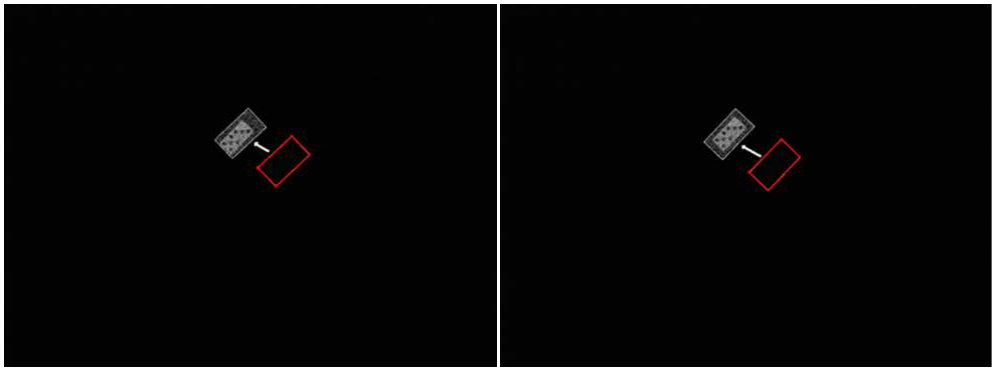

Fig. 5. Domino piece search. a) Landscape image and object recognized for the LINT model. b) Landscape image and object recognized for the BIN model.

Algorithm	**Object model**	**x**	**y**	θ	**t**	**S**	**Eval.**
emCGA (a)	LINT	580	434	317.03	-	50.15	8.32E+04
ESA (a)	LINT	580	428	123.99	-	50.16	4.03E+08
emCGA (b)	BIN	572	425	313.5	5	85.96	9.43E+04
ESA (b)	BIN	572	425	313.5	5	85.96	1.03E+11

Table 5. Results for the comparison of two object models and exhaustive search algorithm.

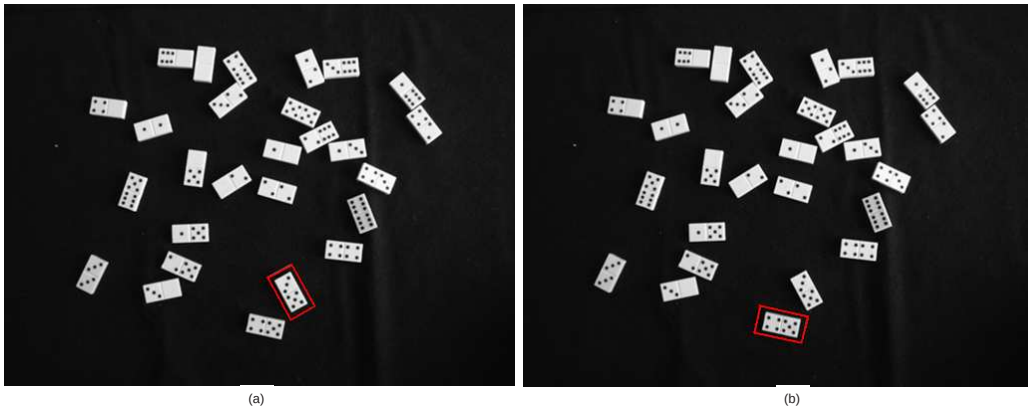

Fig. 6. Complex domino piece search. a) Landscape image and object recognized for the LINT model. b)Landscape image and object recognized for the BIN model.

as the emCGA, can achieve competitive results to an ESA, but with significant smaller computational cost (based on the number of fitness evaluations).

The experiments were developed to evaluate the efficiency of emCGA in situa-

Algorithm	Object model	x	y	θ	t	S	Eval.
emCGA (a)	LINT	575	193	241.64	-	89.11	8.38E+04
ESA (a)	LINT	575	193	241.64	-	89.11	4.03E+08
emCGA (b)	BIN	523	125	192.33	128	93.38	8.85E+04
ESA (b)	BIN	533	193	241.64	128	95.13	1.03E+11

Table 6. Results for the comparison of two object models and exhaustive search algorithm.

tions with several ingredients of real-world problems. Complex input images were used to evaluate the robustness of the object detection method using the emCGA. The utilization of two object models for the reference image and later comparison enables a better exploration of the emCGA's potential.

Results of the first experiment using the LINT model indicate that the emCGA achieved the global maximum with a computational cost of only 0.022% of that of ESA, and 21.1% of the mCGA. Also, mCGA achieved a performance around 20% worse than emCGA. The first experiment with the BIN model showed that the emCGA reaches the global maximum with a computational cost of approximately 0.008% of that of ESA. This performance is similar to the experiment with the LINT model, but the ESA performance demands a much higher number of evaluations. This is due to the fact that, in the BIN model, an additional parameter (t) is included in the chromosome, thus leading to a huge increment in the search space. Even so, emCGA performed well, when compared with the ESA and mCGA (one order of magnitude faster).

In the second experiment with the LINT model no distortion was added. However, the induced search space of the problem has many local maxima, since there are a lot of similar faces. However, differently from the previous experiment, the amplitude of the fitness function, when the reference image matches each face in the landscape image, will lead to local maxima of about the same value. This challenging problem is commonly found in real-world situations. The emCGA was able to find the global maximum with a computational cost of approximately 0.027% of that of ESA. emCGA was, also, 4 times faster than mCGA which, again, performed worse than emCGA. In fact, mCGA found an object (face) similar, but different from the reference image. This result shows that the proposed emCGA, using the LINT model, has a good discriminatory power, capable of capturing underlying details in the images.

The last experiment of the first group was especially devised to verify how the method behaves in the presence of significant distortions in the images, other features present in real-world images. In the first landscape image the object to be recognized was rotated in two axes, and for the second landscape image, in three axes. For both cases, emCGA reached the global maximum with a computational cost of approximately 0.02% of that of ESA, and emCGA converged 4 to 5 times faster than mCGA. Notice that, for the first landscape image mCGA did not found

an acceptable solution for the problem, since θ is rotated around 180 degrees of the expected orientation. Recall that, in our experiments, the chromosome of emCGA encoded only rotation in plane (x, y). Notwithstanding, for both landscape images emCGA was able achieve good results even in the presence of unexpected distortions in the 3D space.

The last set of experiments were devised to evaluate to compare the efficiency of the object models. In the first experiment of this group, it is observed that both algorithms were able to find the optimal solution, since the problem is relatively easy, with a single domino piece. However, using the LINT model less evaluations were needed, than using the BIN model. The reason for this behavior is that the BIN model deals with a much larger search space than the LINT model.

The next experiment of the second group focused on the effect of strong illumination reduction. In this case, both models needed a somewhat similar number of evaluations, although the LINT model found the object rotated around 180 degrees. This result suggests that the emCGA with BIN model seems to be more adequate, than the LINT model, to find objects with low level of illumination.

The final experiment added more complexity to the scene, with several local maxima. Since all domino pieces have a high degree of similarity each other (compared with the neutral background), even when rotated 180 degrees, this problem is, possibly, the most challenging one of all the experiments. For this experiment, both models required about the same computational effort, much lower than ESA. However, the BIN model did not succeed to find the reference object. Instead, it found a similar one. Overall, it is possible to assert that, based on our experiments, the LINT model is a good option to recognize objects with important details, and the BIN model is more robust for images with low illumination. Both models seems to perform equally in the presence of distortions.

Generally speaking, emCGA using both object models was found adequate and efficient for the task of template matching in digital images, although more extensive experiments should be done to evaluate its possible limitations. The can be applied to real-world problems without increasing significantly the complexity of the implementation or the demand for computational resources. Future work will also investigate the utility of emCGA for other real-world problems, as well as comparing it with other similar approaches. Since the nature of emCGA (as well as other CGAs) makes it especially suited for a parallel implementation in reconfigurable hardware, this certainly will be the next step of this research, applying the system to real-time image processing.

References

1. C. Ahn and R. Ramakrishna, Elitism-based compact genetic algorithms, *IEEE Trans Evol Comput* **7** (2003) 367–385.
2. D. Dumitrescu, B. Lazzerini, L.C. Jain and A. Dumitrescu, *Evolutionary Computation*, (CRC Press, Boca Raton, 2000).
3. C.R. Erig Lima and J.M. Rosário, Reconfigurable architecture proposal applied to

mobile robots, *Proc. ABCM Symp Series in Mechatronics* **1** (2004), pp. 68–75.
4. M.K. Felizberto, H.S. Lopes, T.C. Mezzadri and L.V.R. Arruda, An object detection and recognition system for weld bead extraction from digital radiographs, *Comput Vis Image Underst* **102** (2006) 238–249.
5. E.P. Ferlin, H.S. Lopes, C.R.E. Lima and E. Chichaczewski, Reconfigurable parallel architecture for genetic algorithms: application to the synthesis of digital circuits, *Lect Notes Comput Sci* **4419** (2007) 326–336.
6. J. Gallagher, S. Vigraham and G. Kramer, A family of compact genetic algorithms for intrinsic evolvable hardware, *IEEE Trans Evol Comput* **8** (2004) 111–126.
7. G. Harik, E. Cantu-Paz, D.E. Goldberg and B.L. Miller, The gambler's ruin problem, genetic algorithms, and the sizing of populations, *Evol Comput* **7**:3 (1999) 231–253.
8. G. Harik, F. Lobo and D.E. Goldberg, The compact genetic algorithm, *IEEE Trans Evol Comput* **3**:4 (1999) 287–297.
9. J. He and X. Yao, From an individual to a population: an analysis of the first hitting time of population-based evolutionary algorithms, *IEEE Trans Evol Comput* **6** (2002) 495–511.
10. R. Jain, R. Kasturi and B.G. Schunck, *Machine Vision*, (McGraw-Hill, New York, 1995).
11. P. Larrañaga and J.A. Lozano, *Estimation of Distribution Algorithms: A New Tool for Evolutionary Computation*, (Kluwer, Doordrecht, 2002).
12. H.S. Lopes, C.R.E. Lima and N.J. Murata, A configware approach for high-speed parallel analysis of genomic data, *J Circ Syst Comp* **16** (2007) 1–15.
13. H.A. Perlin, H.S. Lopes and T.C. Mezzadri, Particle swarm optimization for object recognition in computer vision, *Lect Notes Comput Sci* **5027** (2008) 11–21.
14. N.K. Ratha and A.K. Jain, Computer vision algorithms on reconfigurable logic arrays, *IEEE Trans Par Distr Syst* **10** (1999) 29–43.
15. A. Rogers and A. Pruegel-Bennett, Genetic drift in genetic algorithm selection schemes, *IEEE Trans Evol Comput* **49** (1999) 298–303.
16. K. Sastr and D.E. Goldberg, On extended compact genetic algorithm, *Proc. Late Breaking Papers in Genetic and Evolutionary Computation Conf.* (2000), pp. 352–359.
17. R.R. Silva, H.S. Lopes and C.R. Erig Lima, A new mutation operator for the elitism-based compact genetic algorithm. *Lect Notes Comput Sci* **4431** (2007) 159–166.
18. D. Thierens, D.E. Goldberg and A. Pereira, Domino convergence, drift and temporal salience structure of problems, *Proc. of IEEE World Congress on Computational Intelligence* (1998), pp. 535–540.
19. S. Tsutsui, Probabilistic model-building genetic algorithms in permutation representation domain using edge histogram, *Lect Notes Comput Sci* **2439** (2002) 224–233.
20. C. Zhou, K. Meng and Z. Qiu, Compact genetic algorithm mutated by bit, *Proc. of the* 4^{th} *World Congr. on Intelligent Control and Automation* **4** (2002), pp. 1836–1839.