Improving Face Detection

Penousal Machado[1], João Correia[1], and Juan Romero[2]

[1] CISUC, Department of Informatics Engineering, University of Coimbra, 3030 Coimbra, Portugal machado@dei.uc.pt, jncor@dei.uc.pt
[2] Faculty of Computer Science, University of A Coruña, Coruña, Spain jj@udc.pt

Abstract. A novel Genetic Programming approach for the improvement of the performance of classifier systems through the synthesis of new training instances is presented. The approach relies on the ability of the Genetic Programming engine to identify and exploit shortcomings of classifier systems, and generate instances that are misclassified by them. The addition of these instances to the training set has the potential to improve classifier's performance. The experimental results attained with face detection classifiers are presented and discussed. Overall they indicate the success of the approach.

Keywords: Face Detection, Haar Cascade

1 Introduction

Object detection systems, in particular face detection, have become a hot topic of research. Applications that employ this kind of systems are becoming widespread. For instance, they can be found in search engines, social networks, incorporated in cameras, or in applications for smart phones. Like in other example-based learning techniques, the datasets employed are vital, not only for attaining competitive performances, but also for correctly assessing the strengths and shortcomings of the classifiers. As such, developing adequate datasets for training, testing and validation becomes a crucial and complex process.

The use of Evolutionary Computation (EC) techniques in the fields of Computer Vision (CV) and Machine Learning (ML) is widespread. Among other applications, EC has been used for digital filters tuning, parameter optimization and image generation. In the field of ML, EC applications include evolving classifier parameters, thresholds, feature selection for classification, the classifier itself, etc. Works such as [17, 7, 1, 14] combine EC, CV and ML aspects.

This paper explores the use of Genetic Programming (GP) to assess and improve classifier's performance through the synthesis of new training examples. More specifically, the current work focus on: (i) assessing classifier's performance, (ii) using evolutionary algorithms to generate new examples, (iii) using the generated examples to boost the performance of the classifier.

We propose a novel generic evolutionary framework for classifier improvement through the synthesis of new training examples. This framework is then

instantiated by combining a GP image generation system with a state of the art face detector [19].

The experimental results show that GP was successful in finding shortcomings of the face detector, generating hundreds of images that were incorrectly classified. They also show that the addition of these images to the training set reduces its shortcomings, promoting detection accuracy, and leading to better classifier's performance.

The paper is organized as follows: Section 2 makes a brief overview of related work; Next, in section 3, we present the proposed framework for classifier's performance; Section 4 describes the experimental setup; The experimental results are presented and analyzed in section 5; Finally, in section 6, overall conclusions are drawn and future research indicated.

2 State of the Art

As previously mentioned, EC has been used in the development and improvement of classifier systems. However, we were unable to find works that match closely the approach proposed in this paper. In this section we make a short overview of approaches that share common features and goals.

Ventura et al.[18] used EC to generate training samples for a Neural Network (NN). The goal was to optimize a computer network routing system. The fitness function was designed to achieve a pre-determined state. The individuals were composed by vectors of control values that represented the state of the network at some point in time. A NN was then trained with the best individuals. This NN was submitted to a series of tests related to the aimed network state. This work represents and attempt to evolve training samples using a GA which is one of the common goals of our work.

The work of Mayer et al. [12] focused in the optimization of NN's training set. It consisted in a Genetic Algorithm Active Selection method. An Active Sampling method generated new data patterns based on the selected training data, in order to enhance the dataset with new information. The GA evolved subsets of training and sampled data, which were used to train NNs. These NNs were assessed by a testing set. The performance in the test phase determined which subset was the best training set to be used. Related to our work, this approach generates new training samples from the existing samples.

Chen et al. [2] proposed a self-adaptive GA to improve face detection systems. It consisted on resampling the face training dataset. The individuals of the GA were encoded as strings containing the pixel intensity values. The individuals were submitted to mutation and recombination operators. Recombination consisted in segmenting two individuals and combining some of the segmented parts. Mutation consisted in the probability of changing illumination, position and angle of the selected segmented parts. The whole process starts with the face training set being employed as an initial population to perform GA operations. The intermediate solutions of each generation were evaluated by a Sparse Network of Winnows (SNoW) classifier [20], trained with the last non face samples.

The classifier output was used to assign fitness of the individuals. The fittest individuals continued to the next generation and the weaker were discarded. The last population was added to the face training set which was used to train a new SNoW classifier. In this case it relates to this paper due to the usage of the classifier output to fitness assignment.

More recently, D. Dubey [3] explored the face detection problem by using NNs, resampling methods and a GA. The images pixel intensity values were considered as individuals. The initial face training set examples were *resampled* by using rotation and scale operations, generating and adding new samples to the original ones. The non-face training set started with white noise images, created by assigning random intensities to each pixel. A NN was trained to discriminate between face and non-face image with the initial sets. The GA was used to evolve the non-face initial set. The output of the NN was used to assign fitness. The non-face image set was updated by randomly selecting non-face individuals that were misclassified during fitness. In each generation a new NN was trained. After ending the GA process, a final NN was trained with the last generation of non-faces and existing face images. The relation to our work lays on the usage of misclassified examples and their inclusion in the training set.

3 The Framework

The proposed framework comprises three main modules: EC engine, Classifier and Supervisor. Figures 1 and 2 present an overview of the framework and the interaction between the EC engine and Classifier, respectively.

The application of this approach involves the following steps:

1. Selection of a positive and negative image set;
2. A Classifier System (CS) is trained based on the positive and negative instances;
3. N independent EC runs are started; The CS is used to classify the generated individuals; Their fitness depends on the results, including intermediate ones, of the classification task;
4. The EC runs stop when a termination criterion is met (e.g., a pre-established number of generations, attaining a fitness value);
5. The set of negative images is updated by adding the evolved images for which the CS and the Supervisor do not agree (e.g. classified as positive by the CS and as negative by the Supervisor)

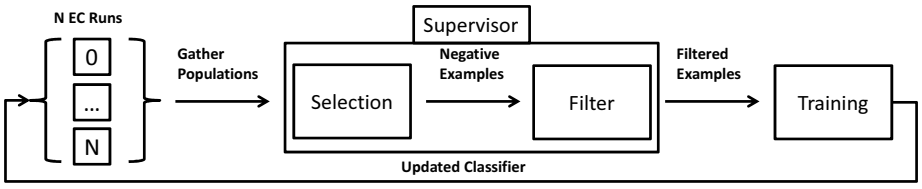

Fig. 1: System overview.

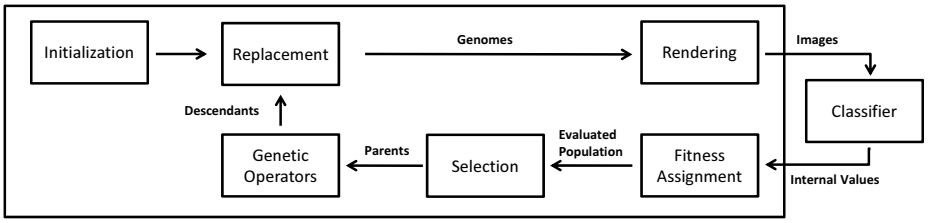

Fig. 2: Evolutionary model and its interaction with the classifier.

6. The process is repeated from step 2 until the boosting criterion is met;

By explaining how this framework is instantiated we will also explain the underlying rationale. In the context of this paper the CS system consists in a Haar Cascade classifier (see Viola et al. [19]) built to detect frontal faces. The code and executables are included in the OpenCV API[3]. This CS approach was chosen due to its state of the art relevance and for its fast classification. This algorithm uses a set of small features in combination with a variant of the Adaboost [4], and is able to attain efficient classifiers. The classifiers assume the form of a cascade of small and simple classifiers that use Haar features [13].

The EC engine used in this experiments is inspired by the works of Sims [15]. It is a general purpose, expression-based, GP image generation engine that allows the evolution of populations of images. The genotypes are trees composed from a lexicon of functions and terminals. The functions include mathematical and logical operations; the terminal set is composed two variables, x and y, and random constant values. The phenotypes are images, rendered by evaluating the expression-trees for different values of x and y, which serve both as terminal values and image coordinates. In other words, to determine the value of the pixel in the (0,0) coordinates one assigns zero to x and y and evaluates the expression-tree. A thorough description of the GP engine can be found in [10].

In the context of this paper, positives are images that contain faces while negatives are images where no face is present. The goal of the EC engine is to evolve images that the CS classifies as faces. To create a fitness function able to guide evolution it is necessary to convert the binary output of the face detector, to one that can provide suitable fitness landscape. This is attained by accessing internal results of the classification task that give an indication of the degree of certainty in the classification. As such, images that are immediately rejected by the classifier will have lower fitness values than those that were close to be classified as possessing a frontal face.

Considering the structure of the selected classifier and through trial and error we developed the following fitness formula:

[3] OpenCV — `http://opencv.willowgarage.com/wiki/`

$$fitness(x) = \sum_{i}^{countstages_x} beststagedifference_x(i) * i + countstages_x * 10 \quad (1)$$

Variables $countstages_x$ and $beststagedifference_x(i)$ are extracted from the face detection algorithm. Variable $countstages_x$, holds the number of stages that image, x, has successfully passed in the cascade of classifiers. The rationale is the following, an image that passes several stages is likely to be closer to being recognized as having a face than one that passes fewer stages. In other words, passing several stages is a pre-condition to being identified as a face image. Variable $beststagedifference_x(i)$ holds the maximum difference between the threshold necessary to overcome stage i^{th} and the value attained by the image at the i^{th} stage. Images that are clearly above the thresholds necessary to pass each stage are preferred over ones that are only slightly above them. Obviously, this fitness function is only one of the several possible ones. Although room for improvement is likely to exist, such improvements are not necessary for the goals of the current paper.

The proposed framework relies on the ability of EC systems to find and exploit the shortcomings of the classifiers to "artificially" increase fitness. The propensity of EC to find "shortcuts" that exploit weaknesses of the fitness assignment scheme is well-known (see, e.g., [16, 17, 11]). Thus, the goal is to evolve false-positives: images that are classified as faces, but that should not have been classified as faces. By adding this false-positives to the negative training set and re-training the CS we wish to correct exploitable flaws of the classifier.

The Supervisor for this experiment is an automatic module that is responsible for gathering all distinct images created during the EC runs. Evolved images that are classified as faces are added to the training set for the next boosting iteration.

4 Experimental Setup

To assess the validity of the proposed approach we performed 30 independent runs of the framework described in the previous section. The framework proposes the use o N independent evolutionary runs, however, we are primarily interested in assessing the contributions that each EC run may bring. Thus, for the scope of this paper we set $N = 1$ and perform 30 independent runs of the the proposed framework. In this section we describe the experimental settings employed in these runs.

4.1 Classifier Training

For training purposes we used the "opencv_haartraining" tool of OpenCV. The relevant classifier parameters are presented in table 1 and were chosen based on the works of Viola [19] and Lienhart [9, 8]. They reflect a compromise between

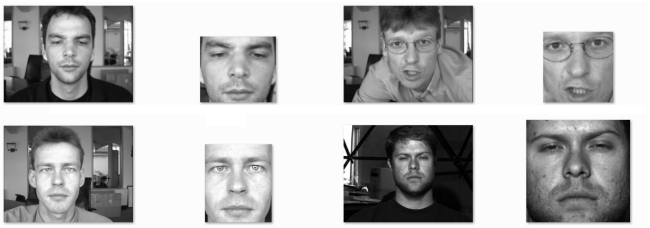

Fig. 3: Examples of cropped positive images.

attaining good classifier's performance and manageable training time. In addition to the training parameters, there are other classifier settings that need to be established. We chose to use the default parameters of OpenCV (see table 2).

The quality of the positive and negative datasets used in training significantly influences the performance of a classifier. It is important to have good positive examples of the object that we are training in order to attain good success rates. For this experiment images from two well-known datasets were used: "The Yale Face Database B" ([5]) and "BioID Face Database"([6]). "The Yale Face Database B" is a dataset with a total 5850 grayscale images with the subjects in diverse positions and light variations. The Bio-ID Face Database dataset has 1521 frontal grayscale images. Each image shows the frontal view of a face of one out of 23 different test persons with various expressions.

We wish to test if the proposed framework contributes to improvements of classifier's performance. Adding different poses has no interest in this context and would make development and analysis harder. As such, we decided to focus exclusively on frontal faces. Although it is easier to develop a good initial classifier, it is likely to make improvements harder, since there is less room for improvement.

Considering this constraint, the total number of available positive examples is 2172. In order to build the ground truth file, the images have to be manually selected and cropped. These cropped images, see figure 3, are the objects that the Haar classifier attempts to discriminate from negative samples. After

Table 1: Haar Training parameters.

Parameter	Setting
features	ALL
Input width	20
Input height	20
Number of stages	14
Number of splits	1
Min Hit rate	0.999
Max False Alarm	0.5
Adaboost Algorithm	GentleAdaboost

Table 2: Classifier parameters.

Parameter	Setting
Window width	20
Window height	20
Scale factor	1.2
Min face width	$0.75 \times inputwidth$
Min face height	$0.75 \times inputheight$

Fig. 4: Examples of negative images.

manually filtering out images that were too dark, or where only part of the face was illuminated, a total of 1905 positive examples, and corresponding cropped versions, was attained.

The negative dataset influences both the training time and test performance. Generally speaking hard and large negative datasets imply longer training times, but also better performance. We employed the "Urtho - Negative face Dataset"[4], which consists of a total of 3019 images of landscapes, objects, drawings, etc. To keep the carnality of the negative and positive datasets balanced we randomly selected 1905 of the Urtho images. A sample is presented in figure 4.

4.2 Genetic Programming Engine

The settings of the GP engine are presented in table 3. The number of generations may appear low, however, preliminary experiments indicated that 50 generations were enough to evolve images classified as faces. Further tests showed that some of the evolutionary runs (23% in the conducted experiments) were unable to evolve such images, but also that increasing the number of generations was inefficient. Therefore, we tackle this problem as follows: if after 50 generations the evolutionary run is unable to find a minimum of 300 images, this run is discarded, and a new evolutionary run with a different random seed is initiated.

4.3 Assessing Classifier's Performance

In order to test the different classifiers, a performance evaluation tool was implemented. It allows loading an image test set, with a ground truth file associated, and a classifier configuration file. The performance is measured in terms of hits (H), misses (M), false alarms (FA), correct (C) and incorrect (I). In order to do this, it loads the parameters and classifier of the configuration file, and perform the face detection. Then it compares the result with the ground truth file. If the result matches or lays within the tolerance area defined by the performance tool parameters, it is a hit. If it lays outside the tolerance area it is counted as

[4] Tutorial haartraining — `http://tutorial-haartraining.googlecode.com/svn/trunk/data/negatives/`

Table 3: Parameters of the GP engine.

Parameter	Setting
Population Size	100
Number of generations	50
Crossover probability	0.8
Mutation operators	sub-tree swap, sub-tree replacement, node insertion, node deletion, node mutation
Initialization method	ramped half-and-half
Initial maximum depth	5
Mutation max tree depth	3
Function set	+, -, *, /, min, max, abs, sin, cos, if, pow, mdist, warp, sqrt, sign, neg
Terminal set	X, Y, scalar and vector random constants

Table 4: Parameters used by the performance tool.

Parameter	Setting
Minimum Window width	20
Minimum Window height	20
Scale factor	1.2
Maximum size difference factor	1.5
Maximum position difference factor	0.3

a false alarm. If no face is detected and a face exists, it is counted has a miss. A positive instance is considered correctly classified if, and only if (i) at least one face was detected and (ii) the regions were the faces were detected match the expected region. In other words, there must be at least one hit and no false alarms. An example follows, if the classifier identifies 2 faces on an image, one in the expected position and the other in an incorrect position, then the instance is considered incorrectly classified. A negative instance is classified as correct if the classifier detects no faces.

The parameters are defined in table 4 and are based on the default parameters of OpenCV's "opencv_performance" tool.

5 Experimental Results

As previously mentioned we performed 30 independent runs of the framework presented in section 3 using the experimental settings described in section 4. As previously mentioned, although the framework proposes the use of several parallel evolutionary runs, we in this test $N = 1$.

Figure 5 displays the evolution of the population average fitness and of the best population individual across generations. In essence this chart shows that in successful runs the GP engine finds images that are classified as faces in few generations. Please notice that runs where the GP was unable to find images

classified as faces were discarded. These runs are useless for improving the classifier's performance since no images would be added to the dataset.

Figure 6 presents examples of images evolved in different evolutionary runs. All of these images have been considered faces by the classifiers. This highlights the shortcomings of the classification system, based on a state of the art classification approach, and further indicates the ability of the GP engine to exploit these shortcomings finding images that are false positives.

Once each evolutionary run ends, the images classified as faces are added to the negative dataset and the classifier is re-trained. This process yields 30 new classifiers. Considering the goals of our research our primary interest is the comparison of the performance of these classifiers with the initial classifier model. For this purpose we consider two validation sets:

- Flickr – 2166 negative images;
- Feret – 902 positive images from Facial Recognition Technology Database[5];

The Flickr image dataset consists in images retrieved from a search in Flickr using the keyword "image" and excluding from the resulting set, images that contain a frontal human face. This process results in a *negative* dataset composed of landscapes, buildings, animals, computer screenshots, varied objects, etc.

The Feret validation set is a *positive* dataset composed by grayscale frontal faces, one face per image with a simple background. The images were manually selected and cropped. The purpose of using this validation dataset is to test the ability of the classifiers in detecting a clear frontal face.

Samples of the validation sets are presented in figure 7.

Table 5 presents a synthesis of the attained results, indicating the performance of the: initial classifier; average performance of the 30 classifiers created using the framework; performance of two of the best framework classifiers found.

[5] The Feret Database – `http://face.nist.gov/colorferet/colorferet.html`

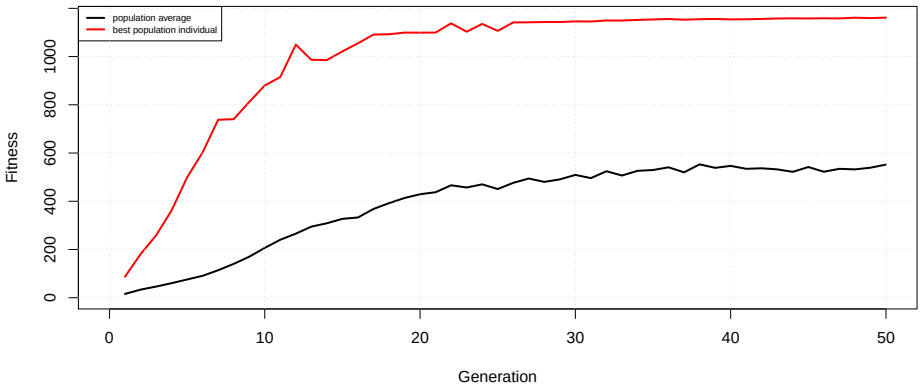

Fig. 5: Evolution of fitness across generations. Results are averages of 30 independent runs.

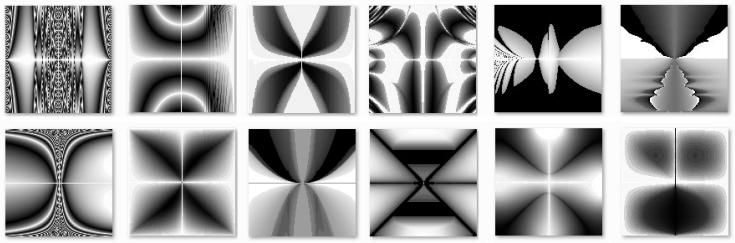

Fig. 6: Examples of evolved images that were classified as faces.

Fig. 7: On the top row, samples of images of the Flickr dataset; On the bottom row samples of the Feret dataset.

Focusing on the comparison of the initial classifier with the average performance of the framework classifiers: the most striking difference in performance is the significant decrease in the number of false alarms which occurs for both validation datasets. On average, for the two datasets, there is a decrease of 25% in the number of false alarms. Adding false positives to the negative training dataset results in classifiers that are more "demanding" than the initial one when it comes to consider the presence of a face in an image. As a consequence, it becomes more robust and precise in the identification, which leads to a decrease in the number of false positives.

The disadvantage is that some face images may go unnoticed. In fact a decrease of the number of hits occurs in the Feret validation dataset (852 vs. 844, which represents a decrease of less than 1%) and, consequently, of the number of misses (50 vs. 58, a 13.8% increase). More importantly, the percentage of correctly identified images (C) increases for both validation datasets. As expected, the improvements of performance are more noticeable in the Flickr dataset, which is composed exclusively of negative images.

It is also important to compare the performance of the best framework classifiers with the performance of the initial models. Table 5 also presents the results of two of the frameworks classifiers that showed increases in performance in both validation datasets. These results demonstrate that it is possible to increase the percentage of correctly identified faces and decrease false alarms without sacrificing the number of hits and misses. Unfortunately, unless it is possible to identify

Table 5: Results attained by the initial classifier and by the framework classifiers in three independent validation datasets. The Flickr dataset is a negative dataset the concept of hits and misses does not apply.

	Flickr		Feret			
Classifier	FA	%C	H	M	FA	%C
Initial	861	73.45	852	50	97	85.59
Average	643.00	78.91	844.00	58.00	77.73	86.12
Classifier 14	581	80.97	852	50	63	88.47
Classifier 30	701	77.75	856	46	64	88.91

which classifiers will show this behavior in validation sets before gathering the validation results, the relevance of this results is limited.

Although in this paper we focus on examining the behavior of the proposed framework, it is important to notice that from a practical perspective we do not need 30 classifiers, we just need one. Further testing is necessary to determine if the performance of these classifiers is generalizable to other validation datasets.

6 Conclusion and future work

A novel evolutionary framework for the improvement of classifier's performance through the synthesis of training examples is presented and discussed. The experimental results attained in two validation datasets show the potential of the approach, demonstrating significant decreases in the number of false alarms and small losses in the number of hits. Additionally, several of the framework classifiers yield better performance in all parameters and for both validation datasets.

Although the results are promising there are several aspects that require further testing and development. Additional testing is necessary to assess if the results attained by the best framework classifiers are generalizable to other validation datasets. The framework anticipates the use of several parallel evolutionary runs and boosting iterations, but the presented results consider only one. Gathering the evolved false positives of all EC runs, adding them all to the negative dataset, and training a single classifier is likely to yield better overall performance. By increasing the number of iterations one forces the EC to focus on different shortcomings of the classifier, which may result in better overall performance.

A final word goes to the supervisor module. Judiciously selecting which images should be added to the negative dataset is likely to contribute to better performances and lower training times. Experiments concerning these aspects are already taking place.

7 Acknowledgments

This research is partially funded by: the Portuguese Foundation for Science and Technology, project PTDC/EIA–EIA/115667/2009; the Spanish Ministry for

Science and Technology, project TIN2008–06562/TIN; Xunta de Galicia, project XUGA–PGIDIT10TIC105008PR.

References

1. Baro, X., Escalera, S., Vitria, J., Pujol, O., Radeva, P.: Traffic Sign Recognition Using Evolutionary Adaboost Detection and Forest-ECOC Classification. Intelligent Transportation Systems, IEEE Transactions on 10(1), 113–126 (2009)
2. Chen, J., Chen, X., Gao, W.: Resampling for face detection by self-adaptive genetic algorithm. In: Pattern Recognition, 2004. ICPR 2004. Proceedings of the 17th International Conference on. vol. 3, pp. 822 – 825 Vol.3 (aug 2004)
3. Dubey, D.: Face detection using genetic algorithm and neural network. International Journal of Science and Advanced Technology (ISSN 2221-8386) 1(6), 104–109 (August 2011)
4. Freund, Y., Schapire, R.E.: A decision-theoretic generalization of on-line learning and an application to boosting. J. Comput. Syst. Sci. 55, 119–139 (August 1997)
5. Georghiades, A.S., Belhumeur, P.N., Kriegman, D.J.: From few to many: illumination cone models for face recognition under variable lighting and pose. IEEE Transactions on Pattern Analysis and Machine Intelligence 23(6), 643–660 (2001)
6. Jesorsky, O., Kirchberg, K.J., Frischholz, R.W.: Robust Face Detection Using the Hausdorff Distance. Computer 2091(June), 90–95 (2001)
7. Krawiec, K., Howard, D., Zhang, M.: Overview of Object Detection and Image Analysis by Means of Genetic Programming Techniques. In: Frontiers in the Convergence of Bioscience and Information Technologies, 2007. FBIT 2007. pp. 779–784 (2007)
8. Lienhart, R., Kuranov, A., Pisarevsky, V.: Empirical analysis of detection cascades of boosted classifiers for rapid object detection. In: Michaelis, B., Krell, G. (eds.) DAGM-Symposium. Lecture Notes in Computer Science, vol. 2781, pp. 297–304. Springer (2003)
9. Lienhart, R., Maydt, J.: An extended set of haar-like features for rapid object detection. In: Proceedings of the 2002 International Conference on Image Processing. pp. 900–903 (1) (2002)
10. Machado, P., Cardoso, A.: All the truth about NEvAr. Applied Intelligence, Special Issue on Creative Systems 16(2), 101–119 (2002)
11. Machado, P., Romero, J., Manaris, B.: Experiments in computational aesthetics: An iterative approach to stylistic change in evolutionary art. In: Romero, J., Machado, P. (eds.) The Art of Artificial Evolution: A Handbook on Evolutionary Art and Music, pp. 381–415. Springer Berlin Heidelberg (2007)
12. Mayer, H., Schwaiger, R.: Towards the evolution of training data sets for artificial neural networks. In: Evolutionary Computation, 1997., IEEE International Conference on. pp. 663 –666 (apr 1997)
13. Papageorgiou, C.P., Oren, M., Poggio, T.: A general framework for object detection. In: Sixth International Conference on Computer Vision. pp. 555–562 (Jan 1998)
14. Sha, S., Jianer, C., Ling, Q., Sanding, L.: Evolutionary mechanism and implemention for recognition of objects in dynamic vision. In: Computer Science Education, 2009. ICCSE '09. 4th International Conference on. pp. 178–182 (2009)
15. Sims, K.: Artificial Evolution for Computer Graphics. ACM Computer Graphics 25, 319–328 (1991)

16. Spector, L., Alpern, A.: Criticism, culture, and the automatic generation of artworks. In: Proceedings of Twelfth National Conference on Artificial Intelligence, pp. 3–8. AAAI Press/MIT Press, Seattle, Washington, USA (1994)
17. Teller, A., Veloso, M.: Algorithm evolution for face recognition: What makes a picture difficult. In: International Conference on Evolutionary Computation. pp. 608–613. IEEE Press, Perth, Australia (1–3 Dec 1995)
18. Ventura, D., Andersen, T., Martinez, T.R.: Using evolutionary computation to generate training set data for neural networks. In: Proceedings of the International Conference on Neural Networks and Genetic Algorithms. pp. 468–471 (1995)
19. Viola, P., Jones, M.: Rapid object detection using a boosted cascade of simple features. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. I–511–I–518 vol.1. Hawaii (2001)
20. Yang, M.H., Roth, D., Ahuja, N.: A snow-based face detector. In: Advances in Neural Information Processing Systems 12. pp. 855–861. MIT Press (2000)