Evolving Visual Features and Detectors

ALVARO GUARDA[1,2], CHRISTOPHE LE GAL[1], AUGUSTIN LUX[1]

[1] Projet PRIMA - Lab. GRAVIR - IMAG
INRIA Rhône-Alpes
655, Av. de l'Europe
38330 Monbonnot St. Martin, France
Alvaro.Guarda,Christophe.Le-Gal,Augustin.Lux@imag.fr

[2] Federal University of Ouro Preto - ICEB - DECOM
Ouro Preto, MG, Brazil

Abstract. This paper addresses the problem of automatic synthesis of visual detectors. We present a method using genetic techniques to learn visual features and a program which combines and integrates the features in non-linear ways. The method is integrated in a face tracking system to generate a variety of new visual perceptual processes.
Keywords: object detection; features and detectors learning; evolutionary learning

1 Introduction

The visual world is rich in possibilities, unpredictable, and complex. In addition, the basic representation, the image, has a low-level representation, is noisy and, due to the projection, does not have a direct, obvious and unique correspondence with the reality to be understood. The knowledge necessary to remove the ambiguities and to control the complexity is huge. For all these reasons and in order to be effective, a computer vision system must be highly specialized, containing task-oriented features and knowledge which are specific to a particular application area. However, it is very hard to realize all the relevant aspects of a specific visual task, and each individual application may require a big amount of development time and the results are not readily applicable to other tasks.

Due to these difficulties, the use of machine learning techniques in computer vision systems is very attractive, and, in fact, it is not new, mainly in pattern recognition. However, their use is frequently either restricted to numeric and parametric learning or based on unrealistic assumptions. The main reason is that most symbolic learning methods are not appropriate for vision problems. These techniques use high-level symbols to represent knowledge and to reason about problems, assume a good segmentation, and that the extracted data are precise, complete and noise-free, in opposition to real data in computer vision.

In this paper, we propose a method based on genetic techniques to learn visual detectors of objects, given by a user. The detectors learned are integrated into a face tracking system.

Genetic Techniques and Related Works

Evolutionary Computation is a family of adaptive searching techniques in a solution space. It is inspired by the simulation of evolution and natural selection, and uses an iterative and stochastic process that operates on a population of individuals. An individual is a solution, like a program, or a representation of a solution, like a string of parameters. The basic process involves firstly the generation and evaluation of an initial population, and secondly an iterative application of evolutionary operations that includes crossover and mutation. The methods for selection of the individuals being used in the evolutionary operations are fitness-proportionate. This means individuals with higher fitness have a higher chance of being selected. Then, the population will tend to converge to groups of individuals with the best fitness (best solutions to the problem). The final solution is usually the best individual which appeared during all iterations. The evaluation of the individuals is done by a mathematical function using several criteria, as the fitness of the solution, the cost of the solution, and so on. Its form depends strongly on the problem to be solved and it can become very complex.

In this work, we use two of this group of techniques: Genetic Algorithms and Genetic Programming. The description above is valid for both of them. The main difference is the representation of the solution, which will affect the evolutionary operations. The choice depends on the problem characteristics.

In a traditional Genetic Algorithm, the solutions are represented as strings of fixed length, representing the individuals. In [11, 8], Holland and Goldberg give a very good explanation of the subject.

Genetic Programming evolves programs to do a specific task which are represented as a parse tree. The language of the resulting programs is defined by a **function** set, the internal nodes, and a **terminal** set, the leaves of the tree. See [15, 16] for a extensive quantity of information.

The last few years have shown growing interest in Evolutionary Computation methods in the domain of image processing and computer vision. Below is a brief discussion of some works resembling this one in some aspect.

Tackett [20] uses Genetic Programming to combine a pre-defined set of intensity and moment-based features in a tree classifier. The features are weighted and combined through linear and nonlinear operations to form the tree classifier. The interesting idea here is the use of Genetic Programming not only for feature selection, but also to weight and combine the features in more complex or expressive ways than usually.

Andre [1] describes a method for the simultaneous evolution of 2-dimensional hit-miss matrices (feature detectors or filters) and an algorithm for using them. The Genetic Algorithms and Genetic Programming techniques are used on binary images to evolve the filters and the algorithms respectively. Each algorithm has 5 associated filters and the possible operations are very simple. There is a pointer to the image, like a cursor, and the operations can move the pointer in the cardinal directions and apply the filters in the pointer position. The filters results are true or false and the others operations are the conditional and the sequence commands. The algorithm returns always true or false. Highlight of the method, regardless of the lack of expressiveness of the language, is the evolution of filters for the discovery of new features.

Johnson et al [13] evolve visual routines to find hands position in silhouettes of persons. Genetic Programming is applied on binary images with high-level operations like Centroid, Bounding-box-top-left-point, Find-bottom-edge, and so on. The result is a program combining these operations that returns a point hitting a hand. The language used is much more expressive than that one described in [1], but the search of the object, a hand, is somewhat blind and not robust to non-conventional positions of the object due the use of binary images and features that don't have any knowledge or models of hands.

Teller and Veloso [21] propose a system that learns visual object recognition programs. It learns a set of recognizers using Genetic Programming and combines them using a tuning weight vector learned or adjusted during its test phase. Gray-scale images are used and the language has high-level operations like variance and average in a window. They present several interesting ideas as the use of a library of internal functions (ADF - Automatically Defined Functions [16]), the evolution of crossover operators, the use of weights and confidence values for the recognizers, the use of more than one evolved recognizer to decide on a confidence for a class, and the incremental capability. However, their approach is to study the problem of general signal-to-symbol conversion, plus they do not use any knowledge of the task. In addition, some questions that still must be worked out are the incremental capability and the confidence values of the recognizers. The system uses all previous examples in addition to the new ones each time it runs, and it is not really incremental. The confidence values returned by the recognizers are somewhat random and therefore rather meaningless.

De Jong et al use Genetic Algorithms for feature selection [22, 2], feature space restructuring [23], and feature construction [3]. They use induction of decision trees for the evaluation function, in a hybrid architecture.

Other than in [1, 13], the experiments in this work use color images, which are neither normalized nor improved in any sense. Indeed, a feature set is neither defined nor selected in advance, like it is done in [20, 23, 3]. We propose the use of fuzzy logic to integrate the learned detectors, improving the incremental capability of the system over that in [21].

In [9] we proposed the use of genetic methods for learning detectors for the classification of objects. The main differences in this current work are 1) the use of color images to improve the invariance to the light level, 2) the coding of the masks, allows a more rich set, 3) the language used in the genetic programming, where topological information was introduced, and 4) its application in a face tracking system that imposed more subtle changes.

2 Method Overview

The aim of our method is to learn object detectors given by a user. The method has two main stages: learning and detection. The learning stage does a supervised induction on a set of training examples. The result is a set of detectors that are integrated in a face tracking system. In this stage, the system can interact with the user to create new examples that can be used to learn new detectors. The detectors are a combination of local features and are represented in a fuzzy logic based language.

Figure 1 shows the architecture of the learning stage. First, a genetic algorithm module learns a set of masks, whose application represents the local features (see section 3). These features are then combined by a genetic programming module as described in section 4. The genetic modules work as was explained in the introduction.

The evaluation of the populations is done by a fitness function that uses a set of training examples (fitness cases), given by the user. Each example is a window enclosing an "object" on the image. See figure 2 for some

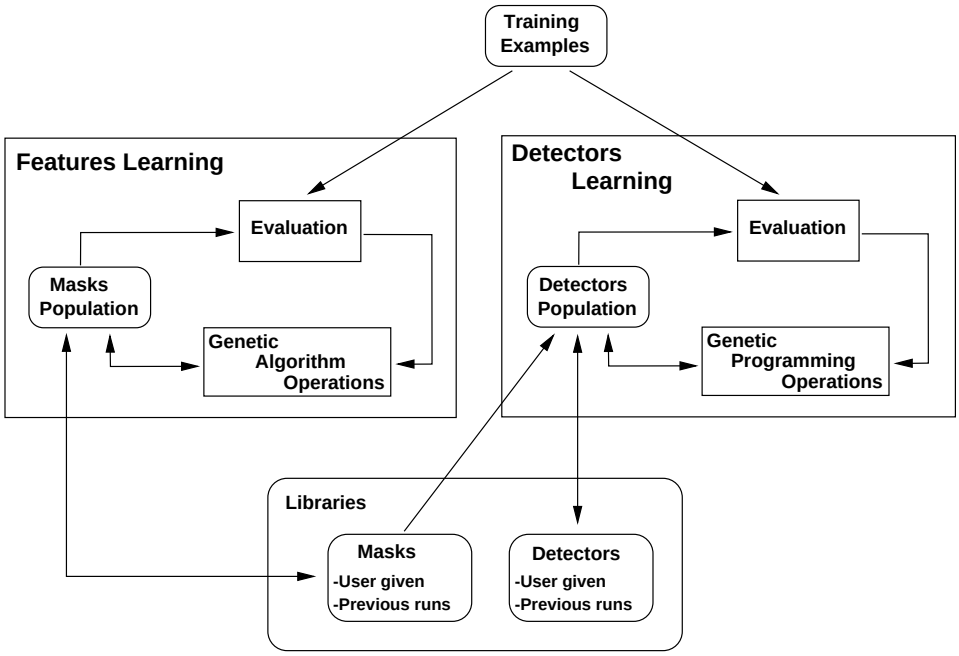

Figure 1: Architecture of the learning stage.

Figure 2: Facial elements of an image used as training examples.

examples.

The representation of the detectors as fuzzy logic rules has the advantage that the knowledge gained can easily be incremented. The system can be run several times with different examples and, at each run, new rules are added to the detectors library.

The method is incremental in two other ways. First, the libraries of masks and detectors evolved in past runs is used to aid the system to converge faster to a new solution, taking advantage of previous work. Second, libraries of masks and detectors entered by an user is a mechanism that allows a specialist to insert knowledge of the application domain.

The detectors in the library can be selected by the supervisor of the tracking system described in section 5.

The method was implemented in the languages Scheme and C on the RAVI system [18, 25]. The implementation was based on the code given in [16].

Improving the robustness to the light variations

A grey-scale image contains several kinds of informations (light, shape, texture, ...). However, they are mixed and, consequently, difficult to exploit correctly. Even if an important part of the information is carried by the grey-scale image, the use of color allows, to a certain extent, for the separation of those informations. Particularly, the use of color allows to improve the robustness to the light level variations [6].

Let us consider the (simplifying) assumption the RGB components of a color are

$$R = R_0 \sum_{p \in LightSources} I_p \cos\theta_p$$

$$G = G_0 \sum_{p \in LightSources} I_p \cos\theta_p$$

$$B = B_0 \sum_{p \in LightSources} I_p \cos\theta_p$$

where (R_0, G_0, B_0) is the real color of the object, $LightSources$ is the set of light sources in the scene, I_p is the

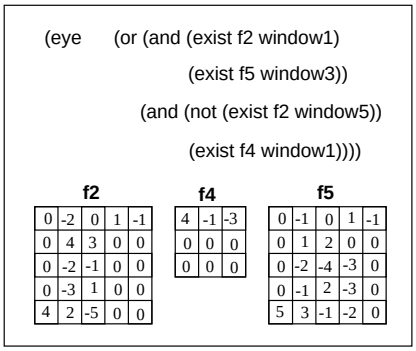

Figure 3: The first rule learned for the eyes and its masks.

intensity of the source p, θ_p is the angle between the normal of the object's surface, and the light rays are coming from the source p. (This hypothesis is sometimes used in computer graphics for a specular surface and white light [7]).

Under this hypothesis, the components

$$u = \frac{R}{R+G+B}; v = \frac{G}{R+G+B}$$

become

$$\begin{aligned} u &= \frac{\sum_{p \in LightSources} R_0 I_p \cos\theta_p}{\sum R_0 I_p \cos\theta_p + \sum G_0 I_p \cos\theta_p + \sum B_0 I_p \cos\theta_p} \\ &= \frac{R_0}{R_0+G_0+B_0} \\ v &= \frac{G_0}{R_0+G_0+B_0} \end{aligned}$$

and are then independent from light variations and object orientation. The (u, v) decomposition is used by Hurlbert and Poggio to perform a robust segmentation [12], and by Schiele and Waibel for gaze tracking [19].

Of course, as pointed out by Luong [17], this assumption is naive, and untrue in presence of shadows. However, under our experimental conditions, the light is diffuse and the objects do not have sharp shadows.

Our method uses u, v, and grey-scale components as input images. The learning stage chooses automatically the components to extract the features.

3 Local Features Learning

A feature is coded as a 2-dimensional mask where each element is in the range of [-5,+5]. Figure 3 shows examples of evolved masks. In this coding scheme, we can represent gradient masks like Robert masks and Prewitt masks and even some small Laplacian masks.

A local feature is the inner product of a mask at a certain point in the image. The result at each pixel is normalized to the range of [0, 1] and represents the degree of matching of the point neighborhood with the mask.

The masks are evolved by a genetic algorithm (GA), as shown in figure 1. In the following subsections we describe the main components of the genetic algorithm.

3.1 Genetic operators.

Genetic operators (mutation and crossover) are used to change the population in searching new masks. We employ operators similar to those of Andre [1], which can be described as follows. As a mask has a 2-dimensional structure and adjacency information is relevant, it is interesting that the evolutionary operators are applied in a 2-dimensional way. A mutation in a mask is done by choosing a random window size (rectangle) and a random position in the mask, then the contents of the window in the mask are randomly changed. Crossover involves two masks. First, window size and the positions of the window in each mask are randomly chosen. Then, the contents of the window in one mask are exchanged with the contents of the other.

3.2 Fitness function.

The evaluation of the individuals is done by a fitness function. As fitness is an error function, a perfect mask has $fitness = 0$.

Let I be an image, W the window enclosing an object and M a mask. The fitness of the mask M for image I is

$$fitness_M(I) = \sum_{pixel_{ij} \in I-W} \text{greater}(M(pixel_{ij}), \max_W M),$$

that is, the number of points in $I - W$ for which the response of the mask is higher than its maximum response in W (greater(x, y) = 1 if $x > y$, 0 otherwise). The total fitness of the mask is the sum of the mask fitness for all the images of the training base. Thus, masks obtaining good marks are those able to distinguish a feature of an object from the rest of the image (a perfect mask will always give his best marks to one of the object pixels).

4 Combination of features

Learning the detectors is done by a genetic programming module, as shown in figure 1. A detector is a combination of several local features (masks). This combination allows for the use of spatial information to remove ambiguities and to achieve more reliable results. Figure 4 presents the results of the combination of two masks for the mouth. The masks alone have mediocre results, but the use of spatial information allows the genetic programming to generate a detector whose highest response is on the mouth center.

The detectors are represented as fuzzy logic rules that state logical relations between the local features (mask convolution). Figure 3 shows a rule learned for the *eyes* and their masks. In the next subsections we describe the

Figure 4: Combination of two masks.

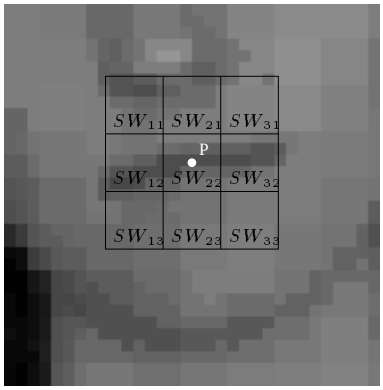

Figure 5: Interpretation of a detector application.

meaning of the language and detail the main points of the genetic programming.

4.1 Language used by genetic programming.

The language of the programs evolved by genetic programming is defined by a set of **functions**, the internal nodes, and a set of **terminals** the leaves of the tree.

Table 1 summarizes the functions of the language showing their type, arity, argument types, and definition. These functions are traditional operations on *fuzzy sets* and are assumed to be within the real interval [0, 1]. More detailed information about fuzzy sets and logic can be found in [24, 14].

A detector is applied to a point in the image. It considers a neighborhood of this point as a window split into 9 areas or sub-windows.

The function *exist* characterizes the presence of the feature defined by $Mask$ on the sub-window $Area$.

$$exist(Mask, Area) = max_{pixel\_\in\_Area\_}Mask(pixel)$$

The detector then verifies spatial relations between the features. This language allows the construction of detectors of the kind

$$Mouth = (AND(exist\_f_1\_area_{11})(exist\_f_2\_area_{31}))$$

This rule says "P is the center of the mouth only if the feature f_1 is present in the sub-window $area_{11}$ and the feature f_2 is present in the sub-window $area_{31}$", as shown in figure 5.

4.2 Genetic operators.

As with genetic algorithms, genetic operators are used to evolve the population of detectors in searching for new solutions. Detectors are represented as expression trees and the application of the genetic operators on them is similar to the conventional genetic programming [15, 16].

4.3 Fitness function.

A *fitness function* evaluates the population of detectors. It measures how well each individual works as a detector. The requirements for a good detector are sensitivity (ability of an individual to detect a given region of the image), specificity (ability to reject the other regions of the image), and efficiency. So, the fitness function for a detector is

$$f_{detector} = f_{sensitivity} + f_{specificity} + f_{efficiency}.$$

As all theses functions are error functions, the lower the response, the better the individual.

The component $f_{sensitivity}$ represents the detector's ability to detect the center of the object. A high response of the detector on the center of the object characterizes high sensitivity (the response of a perfect detector (that is, a detector for which $f_{sensitivity}$ = 0) on the object center would be 1.0). Thus:

$$f_{sensitivity} = 1 - resp(objectCenter)$$

The component $f_{specificity}$ represents the detector's ability to reject all pixels of the image except the center of the object (that is the pixels in $Image - objectCenter$).

A low maximum response of the detector on $Image - objectCenter$, characterizes high specificity (the response of a perfect detector ($f_{specificity} = 0$) on all the pixels of $Image - objectCenter$ would be 0.0). Thus:

$$f_{specificity} = \max_{pixel \in Image - objectCenter}(resp(pixel))$$

Table 1: The function set.

Name	Type	Arity	Argument Types	Definition
not	fuzzy	1	fuzzy	$\text{not}(x) = 1 - x$
and	fuzzy	2	fuzzy, fuzzy	$\text{and}(x, y) = \min(x, y)$
or	fuzzy	2	fuzzy, fuzzy	$\text{or}(x, y) = \max(x, y)$

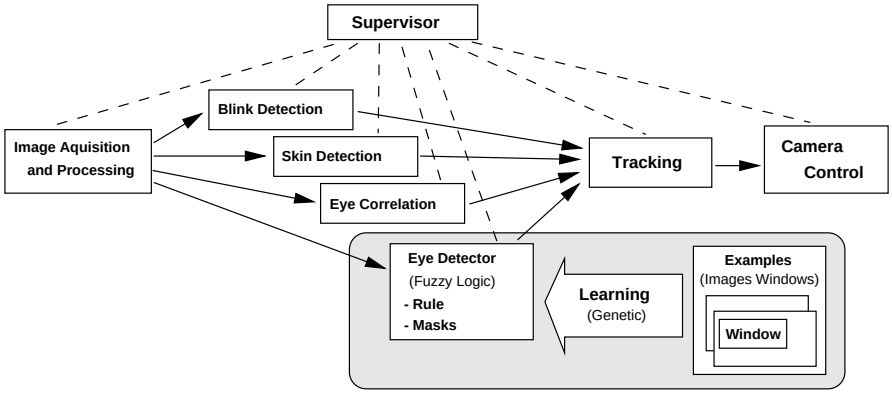

Figure 6: General scheme of the tracking system.

The measure of efficiency,

$$f_{efficiency} = size\_of\_the\_parsing\_tree$$

, is related to the parsimony and simplicity of the individual. In addition, it is simple and easy to calculate.

The two other components of the fitness function use the same set of training examples (fitness cases) as the genetic algorithm.

5 Application: face tracking

Unsurprisingly, robust and fast visual face tracking is a difficult task; there is no simple model for faces under changing illumination and different backgrounds. As there seems to be no single algorithm to solve this problem it is an interesting idea - proposed and pursued in [4, 5, 10] - to use several algorithms, implemented as distinct processes, detecting simple face features, and to combine their results using a supervisor. Based on the global result, the supervisor decides to activate or deactivate processes. For instance, a correlation operator for eye detection may only be applied if a good estimation for eye position exists. When correlation fails, a back-up estimator may be called using skin color.

In such a system, it is useful to have a large number of different processes, such as detectors of the eyes, mouth, nose, ears, or other salient structures. We propose the use of the proposed method to generate such detectors automatically from example pictures.

Generating detector functions automatically from examples is particularly useful in the context of multi-process tasks as described above, because it can operate continuously inside the working system, improving its competence and performance.

The global structure of the face tracking system is shown in figure 6; this figure shows how newly generated detectors connect with the rest of the system, and which data are used to generate detectors.

5.1 Experimental Results

In our experiments, we used the proposed method to learn detectors for the eyes. The experiments have been carried out on a sequence of 100 images. Ten percent of the images were used as training data. The experiments consisted in running the learning module on the training data, and then, applying the learned detectors to the detection of the eyes on the images sequence.

Contribution of Color

Figure 7 shows the evolution of the learning process for both, using color and only grey-scale components. Each curve on the graph represents the evolution average on 10 runs; the used value is the fitness of the best. We can see that when color information is used, convergence is slower, but eventually achieves better results.

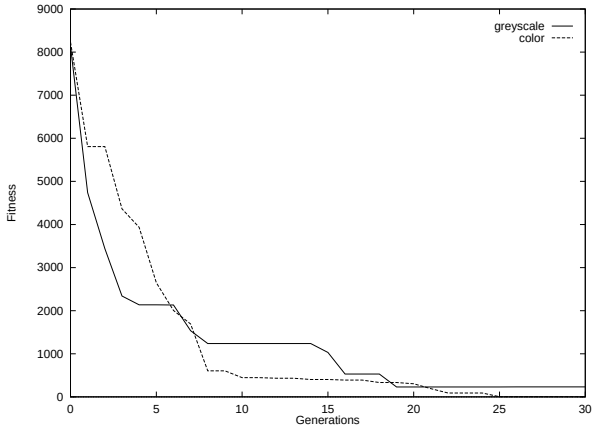

Figure 7: Learning behavior.

Figure 8: Detector response on an image

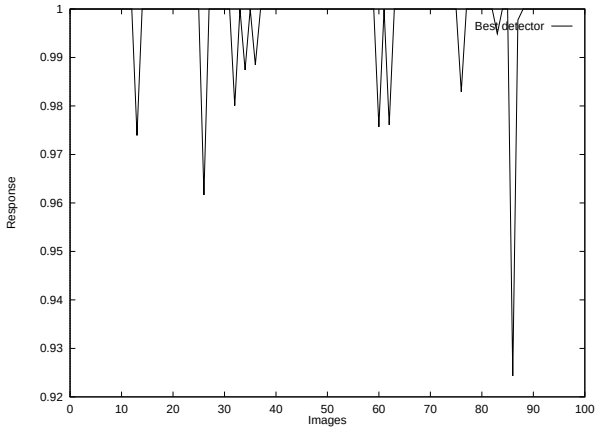

Figure 9: Results of the best detector on the images sequence.

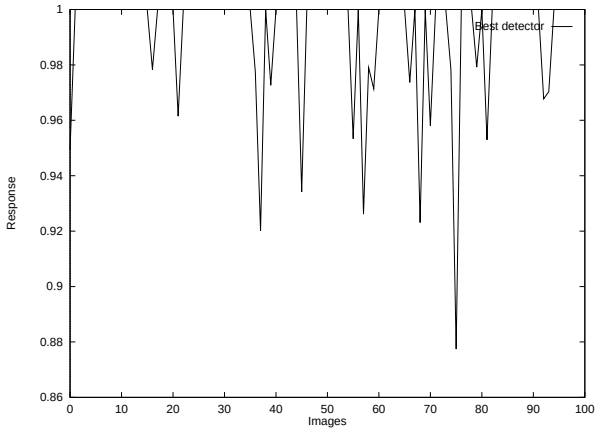

Figure 10: Results on a different sequence.

Best Detector Results on the test set

In this experiment, the genetic algorithm module was used with a population of 500 masks (size between 3x3 and 7x7 pixels) on 50 generations. The genetic programming was used with a population of 600 detectors on 300 generations. The best detector found had a perfect result on the training set images only. Figure 9 shows the response (see figure 8 for an example of a response image) of this detector on the whole sequence. A response lower than 1 means the maximum response was not on the eyes. As shown in figure 9, the maximum response is almost always on the eyes; even if this is not the case, the eyes have one of the best responses. When integrated to the tracking system, a Kalman filter suppresses the errors of the detector.

The results of the detector application on another sequence acquired under different conditions (different light and background) is acceptable, too, as can be seen in figure 10.

6 Conclusions

In this paper we have presented a method to learn visual features and detectors. The method is appearance-based: it is based on images only, and not on abstract models like geometrical models. The learning paradigm is a supervised induction on a set of examples and uses evolutionary computation techniques. A genetic algorithm is used to evolve visual features. Genetic programming evolves detectors combining and integrating the features in non-linear ways, in a language based on fuzzy logic.

At the current stage of the implementation all examples are given by a user. In the future we will allow a supervisor to control the learning module and give new examples to it.

As with other visual perceptual processes, the results of the learned detectors are not perfect. However, their integration and cooperation allows us to improve the global performance of the system.

Acknowledgments

The first author is partially supported by Brazilian Government Fellowship/CAPES. We thank Daniela Hall that kindly allowed us to use her image base and tracking system (described in [10]).

References

[1] D. Andre. Automatically defined features - the simultaneous evolution of 2-d feature detectors and an algorithm for using them. In K. Kinnear, editor, *Advances in Genetic Programming*, chapter 23. MIT Press, Cambridge, Massachusetts, 1994.

[2] J. Bala, K. DeJong, J. Huang, H. Vafaie, and H. Wechsler. Visual routine for eye detection using hybrid genetic architectures. In *Proc. of International Conference on Pattern Recognition*, 1996.

[3] J. Bala, K. DeJong, J. Huang, H. Vafaie, and H. Wechsler. Using learning to facilitate the evolution of features for recognizing visual concepts. In *Special Issue of Evolutionary Computation - Evolution, Learning, and Instinct: 100 Years of the Baldwin Effect*, 1997.

[4] J. Coutaz, F. Bérard, and J.L. Crowley. Coordination of perceptual processes for computer mediated communication. In *Proc. of the Second International Conference on Automatic Face and Gesture Recognition*, 1996.

[5] J.L. Crowley. Integration and control of reactive visual processes. *Robotics and Autonomous Systems*, 16:17–27, 1995.

[6] V. Colin de Verdière. Reconnaissance d'objets par leurs statistiques de couleurs. Master's thesis, Institut National Polytechnique de Grenoble, June 1996. projet de D.E.A.

[7] J.D. Foley and A. Van Dam. *Fundamentals of interactive computer graphics*. Addison-Wesley, 1982.

[8] D.E. Goldberg. *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison-Wesley, Reading, MA, 1989.

[9] A. Guarda and A. Lux. Apprentissage de détecteurs visuels. In *11ème Congrès Reconnaissance des Formes et Intelligence Artificielle*, 1998.

[10] D. Hall. Recognition of facial expressions with principle components space. Master's thesis, University of Karlsruhe, April 1998.

[11] J.H. Holland. *Adaptation in Natural and Artificial Systems*. The University of Michigan Press, Ann Arbor, 1975.

[12] A. Hurlbert and T. Poggio. A network for image segmentation using color. In *Proc. of Denver Conference on Neural Networks*, 1988.

[13] M.P. Johnson. Evolving visual routines. Master's thesis, MIT, August 1995.

[14] A. Kandel. *Fuzzy Mathematical Techniques with Applications*. Addison-Wesley, 1986.

[15] J.R. Koza. *Genetic Programming: On the Programming of Computers by Means of Natural Selection*. MIT Press, Cambridge, MA, 1992.

[16] J.R. Koza. *Genetic Programming II: Automatic Discovery of Reusable Programs*. MIT Press, Cambridge, MA, 1994.

[17] Q.-T. Luong. Color in computer vision: a review. *Traitement du Signal*, 8(1):3–34, 1995. in french.

[18] A. Lux and B. Zoppis. An experimental multilanguage environment for the development of intelligent robot systems. In *Proc. of 4th International Symposium on Intelligent Robotic Systems*, 1997.

[19] B. Schiele and A. Weibel. Gaze tracking based on face color. In *Proc. of International Workshop on Face and Gesture Recognition*, 1995.

[20] W.A. Tackett. Genetic programming for feature discovery and image discrimination. In S. Forrest, editor, *Proc. of the Fifth International Conference on Genetic Algorithms*. Morgan Kauffman, 1993.

[21] A. Teller and M. Veloso. Pado: Learning tree structured algorithms for orchestration into an object recognition system. Technical report, Carnegie-Mellon University, Pittsburgh, PA, February 1995.

[22] H. Vafaie and K. DeJong. Robust feature selection algorithms. In *Proc. of the International Conference on Tools with Artificial Intelligence*. IEEE Computer Soc. Press, 1993.

[23] H. Vafaie and K. DeJong. Genetic algorithms as a tool for restructuring feature space representation. In *Proc. of the International Conference on Tools with Artificial Intelligence*. IEEE Computer Soc. Press, 1995.

[24] L.A. Zadeh. Fuzzy sets. *Information and Control*, 8:338–353, 1965.

[25] B. Zoppis. *Outils pour l'Intégration et le Contrôle en Vision et Robotique Mobile*. PhD thesis, INPG, France, May 1997.