Human-Robot Interaction based on Haar-like Features and Eigenfaces

José Barreto*, Paulo Menezes† and Jorge Dias‡
Institute of Systems and Robotics- University of Coimbra
Polo II, 3020-229 Coimbra - Portugal
Email: *jcbar@alumni.deec.uc.pt,† paulo@isr.uc.pt,‡ jorge@isr.uc.pt

***Abstract*— This paper describes a machine learning approach for visual object detection and recognition which is capable of processing images rapidly and achieving high detection and recognition rates. This framework is demonstrated on, and in part motivated by, the task of human-robot interaction. There are three main parts on this framework. The first is the person's face detection used as a preprocessing system to the second stage which is the recognition of the face of the person interacting with the robot, and the third one is the hand detection. The detection technique is based on Haar-like features introduced by Viola et al. [1] and then improved by Lienhart et al. [2]. The eigenimages and PCA [3] are used in the recognition stage of the system. Used in real-time human-robot interaction applications the system is able to detect and recognise faces at almost 3 frames per second in a conventional 450MHz Intel Pentium II.**

I. Introduction

This paper brings together different techniques to construct a framework for robust and rapid people learning, tracking and real-time recognition in a human-robot interaction environment. The context of this work was the creation of a system that during a guided visit, for example to one of the labs of Institute of Systems and Robotics, could ensure that a robot equipped with a camera keeps interacting with the right person.

Toward this end it was constructed a Real-time face recognition system with a preprocessing stage based on a rapid frontal face detection system using Haar-like features introduced by Viola et al. [1] and improved by Lienhart et al. [2], [4].

The detection technique is based on the idea of the wavelet template [5] that defines the shape of an object in terms of a subset of the wavelet coefficients of the image. Like Viola et al. [1] we use a set of features which are reminiscent of Haar Basis functions. Anyone of these Haar-like features can be computed at any scale or location in constant time using the integral image representation for images. In spite of having equivalent face detection and false positive rates to the best published results [6], [7], [8], this face detection system distinguishes from previous approaches [9] in its ability to detect faces extremely rapidly.

The face recognition system is based on the eigenfaces method introduced by Turk et al. [10]. Eigenvector-based methods are used to extract low-dimensional subspaces which tend to simplify tasks such as classification. The Karhunen-Loeve Transform (KLT) and Principal Components Analysis (PCA) are the eigenvector-based techniques we used for dimensionality reduction and feature extraction in automatic face recognition.

The built system, that will be used in a human-robot interaction application, is able to robustly detect and recognise faces at approximately 3 frames per second in a conventional 450MHz Intel Pentium II. In the same machine the hand detector application achieves a frame rate of 5.4 frames per second.

This article is structured as follows: Sections II, III and IV presents to the face detection mechanism that uses classifiers based on Haar-like features. Section V refers to the eigenimage based recognition of faces. Section VI presents the architecture of the on-line face recognition system whose results are presented on section VII. In this latter section some real data results are presented where it can be seen that multiple faces are detected in images but only one is recognised as the interacting one. The results of the application of the Haar-like feature based classifier applied to hand detection are also shown.Section VIII concludes this article.

II. Features

The main purpose of using features instead of raw pixel values as the input to a learning algorithm is to reduce the in-class variability while increasing the out-of-class variability compared to the raw data and thus making classification easier. Features usually encode knowledge about the domain, which is difficult to learn from the raw and finite set of input data. A very large and general pool of simple Haar-like features combined with feature selection therefore can increase the capacity of the learning algorithm. The speed of feature evaluation is also a very important aspect since almost all object detection algorithms slide a fixed-size window at all scales over the input image. As we will see, Haar-like features can be computed at any position and any scale in the same constant time as only 8 table lookups are needed.

Our feature pool was inspired by the over-complete Haar-like features used by Papageorgiou et al. in [5], [11] and their very fast computation scheme proposed by Viola et al. in [1] improved by Lienhart et al. in [2]. More specifically, we use 14 feature prototypes [2] shown in Fig. 1 which include 4 edge features, 8 line features and 2 center-surround features.

These prototypes are scaled independently in vertical and horizontal direction in order to generate a rich, overcomplete set of features.

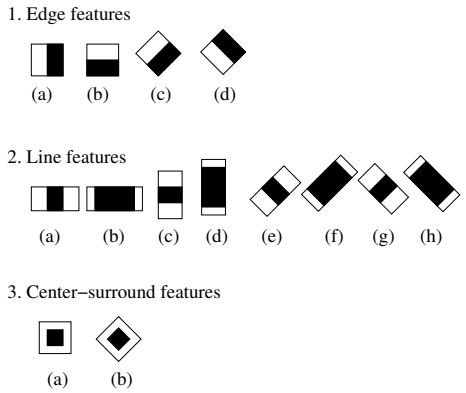

Fig. 1. Feature prototypes of simple Haar-like features center-surround features. The sum of the pixels which lie within the white rectangles are subtracted from the sum of pixels in the black rectangles.

Let us assume that a rectangle of pixels, with top left corner (x, y), width w, height h and orientation $\alpha \in \{0^o, 45^o\}$. This rectangle is inside a window and specified by the tuple $r = (x, y, w, h, \alpha)$ with a pixel sum denoted by $RecSum(r)$. The set of used features have the form:

$$f = \omega_1 \cdot RecSum(r_1) + \omega_2 \cdot RecSum(r_2) \quad (1)$$

where the weights $\omega_1, \omega_2 \in \mathbb{R}$ are used to compensate the difference in area size between the two rectangles r_1 and r_2.

Note that the line features can be calculated by two rectangles only. Here, it is assumed that the first rectangle r_1 encompasses the black and white rectangle and the second rectangle r_2 represents the black area. For instance, line feature $(2a)$ with total height of 2 and width of 6 at the top left corner (5,3) can be written as

$$f = RecSum(5, 3, 6, 2, 0^o) + 3 \cdot RecSum(7, 3, 2, 2, 0^o). \quad (2)$$

Given that the base resolution of the detector is 24×24, the exhaustive set of rectangle features is quite large, over $117,000$ [2]. Note that unlike the Haar basis, the set of rectangle features is overcomplete.

A. Fast Feature Computation

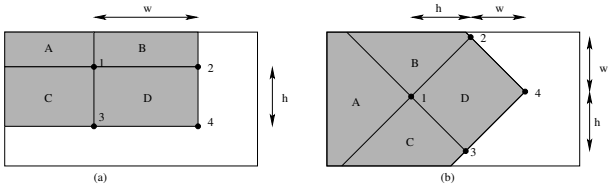

Fig. 2. The sum of the pixels within rectangle D can be computed with four array references. The value of the integral image at location 1 is the sum of the pixels in rectangle A. The value at location 2 is $A + B$, at location 3 is $A + C$, and at location 4 is $A + B + C + D$. The sum within D can be computed as $4 + 1 - (2 + 3)$.

Rectangle features can be computed very rapidly and in constant time for any size by means of two auxiliary images. For upright rectangles the auxiliary image is the *Integral Image* $II(x, y)$. $II(x, y)$ is defined as the sum of the pixels of the upright rectangle ranging from the top left corner at $(0, 0)$ to the bottom right corner at (x, y) (Fig. 2a) [1]:

$$II(x, y) = \sum_{x' \le x, y' \le y} I(x', y'). \quad (3)$$

It can be calculated within a single pass over all pixels from left to right and top to bottom by means of

$$\begin{aligned} II(x, y) &= II(x, y-1) + II(x-1, y) + I(x, y) \\ &\quad -I(x-1, y-1), \end{aligned} \quad (4)$$

with

$$II(-1, y) = II(x, -1) = 0.$$

Based on (3) and (4) the pixel sum of any upright rectangle $r = (x, y, w, h, 0)$ can be determined by four table lookups (see also Fig. 2a):

$$\begin{aligned} RecSum(r) &= II(x, y) + II(x+w, y+h) \\ &\quad -II(x, y+h) - II(x+w, y). \end{aligned} \quad (5)$$

For 45° rotated rectangles the auxiliary image is defined as the *Rotated Integral Image* $RII(x, y)$. It gives the sum of the pixels of the rectangle rotated by 45^o with the right most corner at (x, y) and extending till the boundaries of the image (see Fig. 2b):

$$RII(x, y) = \sum_{x' \le x, x' \le x - |y' - y|} I(x', y') \quad (6)$$

It can be calculated with two passes over all pixels. The first pass from left to right and top to bottom and the second pass from the right to left and bottom to top [2].

From this the pixel sum of any rotated rectangle $r = (x, y, w, h, 45^o)$ can be determined by four table lookups (see Fig. 2b):

$$\begin{aligned} RecSum(r) &= RII(x+w, y+w) + IIR(x-h, y+h) \\ &\quad -IIR(x, y) - IIR(x+w-h, y+w+h) \end{aligned} \quad (7)$$

It becomes clear that the difference between two rectangular sums can be computed in eight references.

III. Learning Classification Functions

Given a feature set and a training set of positive and negative sample images, any number of machine learning approaches could be used to learn a classification function. A variant of AdaBoost [12] is used both to select a small set of features and train the classifier [13]. In its original form, the AdaBoost learning algorithm is used to boost the classification performance of a simple (sometimes called weak) learning algorithm. Recall that there are over 117,000 rectangle features associated with each image 24×24 sub-window, a number far larger than the number of pixels. Even though each feature can be computed very efficiently, computing the complete set is prohibitively expensive. The main challenge is to find a very small number of these features that can be combined to form

• Given example images $(x_1, y_1), ..., (x_n, y_n)$ where $y_i = 0, 1$ for negative and positive examples respectively.

• Initialise weights $w_{1,i} = \frac{1}{2m}, \frac{1}{2l}$ for $y_i = 0, 1$, respectively, where m and l are the number of negatives and positives respectively.

• For $t = 1, ...T$:

1. Normalise the weights,

$w_{t,i} \leftarrow \frac{w_{t,i}}{\sum_{j=1}^{n} w_{t,j}}$

so that w_t is a probability distribution.

2. For each feature, j, train a classifier h_j which is restricted to using a single feature. The error is evaluated with respect to w_t, $\epsilon_j = \sum_i w_i |h_j(x_i) - y_i|$.

3. Choose the classifier, h_t, with the lowest error ϵ_t.

4. Update the weights:

$w_{t+1,i} = w_{t,i} \beta_t^{1-e_i}$

where $e_i = 0, 1$ if the example x_i is classified correctly or incorrectly, respectively, and $\beta_t = \frac{\epsilon_t}{1-\epsilon_t}$.

• The final strong classifier is:

$$h_j(x) = \begin{cases} 1 & \sum_{t=1}^{T} \alpha_t h_t(x) \geq \frac{1}{2} \sum_{t=1}^{T} \alpha_t \\ 0 & otherwise \end{cases}$$

where $\alpha_t = \log \frac{1}{\beta_t}$.

TABLE I

THE ADABOOST ALGORITHM FOR CLASSIFIER LEARNING. EACH ROUND OF BOOSTING SELECTS ONE FEATURE FROM THE 117,000 POTENTIAL FEATURES.

an effective classifier. In support of this goal, the weak learning algorithm is designed to select the single rectangle feature which best separates the positive and negative examples. For each feature, the weak learner determines the optimal threshold classification function, such that the minimum number of examples are misclassified. A weak classifier $h_j(x)$ thus consists of a feature f_j, a threshold θ_j and a parity p_j indicating the direction of the inequality sign:

$$h_j(x) = \begin{cases} 1 & p_j f_j(x) < p_j \theta_j \\ 0 & otherwise \end{cases} \quad (8)$$

here x is a 24×24 pixel sub-window of an image. See Table I for a summary of the boosting process.

IV. CASCADE OF CLASSIFIERS

This section describes an algorithm for constructing a cascade of classifiers [1] which achieves increased detection performance while radically reducing computation time. The key insight is that smaller, and therefore more efficient, boosted classifiers can be constructed which reject many of the negative sub-windows while detecting almost all positive instances. Simpler classifiers are used to reject the majority of subwindows before more complex classifiers are called upon to achieve low false positive rates.

A cascade of classifiers is degenerated decision tree where at each stage a classifier is trained to detect almost all objects of interest (frontal faces or hands) while rejecting a certain fraction of the non-object patterns [1] (see Fig. 3).

Each stage was trained using the Adaboost algorithm (see Table I). Adaboost is a powerful machine learning algorithm and it can learn a strong classifier based on a (large) set of weak classifiers by re-weighting the training samples. At each round of boosting is added the feature-based classifier that best classifies the weighted training samples. With increasing stage number, the number of weak classifiers, which are needed to achieve the desired false alarm rate at the given hit rate, increases (for more detail see [1]).

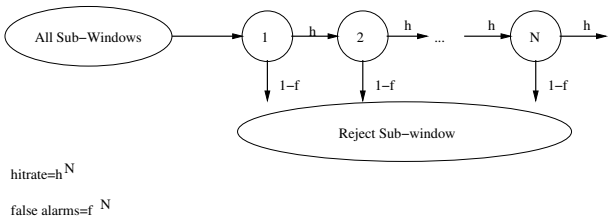

Fig. 3. Cascade of Classifiers with N stages. At each stage a classifier is trained to achieve a hit rate of h and a false alarm rate of f.

V. FACE RECOGNITION USING EIGENFACES

The face recognition system is based on eigenspace decompositions for face representation and modelling. The learning method estimates the complete probability distribution of the face's appearance using an eigenvector decomposition of the image space. The face density is decomposed into two components: the density in the principal subspace (containing the traditionally-defined principal components) and its orthogonal complement (which is usually discarded in standard PCA) [3].

A. Principal Component Analysis (PCA)

Given a training set of $W \times H$ images, it is possible to form a training set of vectors $\mathbf{x}^T$, where $\mathbf{x} \in \mathbb{R}^{N=W*H}$. The basis functions for the Karhunen Loeve Transform (KLT) are obtained by solving the eigenvalue problem:

$$\Lambda = \Phi^T \Sigma \Phi \quad (9)$$

where Σ is the covariance matrix, Φ is the eigenvector matrix of Σ and Λ is the corresponding diagonal matrix of eigenvalues λ_i. In PCA, a partial KLT is performed to identify the largest eigenvalues eigenvectors and obtain a principal component feature vector $\mathbf{y} = \Phi_M^T \tilde{\mathbf{x}}$, where $\tilde{\mathbf{x}} = \mathbf{x} - \bar{\mathbf{x}}$ is the mean normalised image vector and Φ_M is a submatrix of Φ containing the principal eigenvectors. PCA can be seen as a linear transformation $\mathbf{y} = T(\mathbf{x})$: $\mathbb{R}^N \rightarrow \mathbb{R}^M$ which extracts a lower-dimensional subspace of the KL basis corresponding to the maximal eigenvalues. These principal components preserve the major linear correlations in the data and discard the minor ones.

Using the PCA it is possible to form an orthogonal decomposition of the vector space $\mathbb{R}^N$ into two mutually exclusive and complementary subspaces: the feature space $F = \{\phi_i\}_{i=1}^{M}$ containing the principal components and its orthogonal complement $\bar{F} = \{\phi_i\}_{i=M+1}^{N}$. The $\mathbf{x}$ component in the orthogonal subspace $\bar{F}$ is the *distance-from-feature-space* while the component which lies in the feature space F is referred to as the "distance-in-feature-space" (DIFS) [3]. Fig. 4 presents a prototypical example of a distribution embedded entirely in F. In practice there is always a signal component in $\bar{F}$ due to the minor statistical variabilities in the data or simply due to the observation noise which affects every element of x.

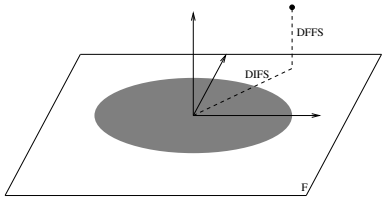

Fig. 4. Decomposition into the principal subspace F and its orthogonal complement $\bar{F}$ for a Gaussian density

The reconstruction error (or residual) of the eigenspace decomposition (referred to as the "distance-from-feature-space" or DFFS in the context of the work with eigenfaces [10]) is an effective indicator of similarity. This detection strategy is equivalent to matching with a linear combination of eigentemplates and allows for a greater range of distortions in the input signal (including lighting, and moderate rotation and scale).

The DFFS can be thought as an estimate of a marginal component of the probability density and a complete estimate must also incorporate a second marginal density based on a complementary *distance-in-feature-space* (DIFS). Using these estimates the problem of face recognition can be formulated as a maximum likelihood estimation problem. The likelihood estimate can be written as the product of two marginal and independent Gaussian densities corresponding to the principal subspace F and its orthogonal complement $\bar{F}$:

$$\hat{P}(\mathbf{x}) = P_F(\mathbf{x}) \cdot \hat{P}_{\bar{F}}(\mathbf{x}) \tag{10}$$

where $P_F(\mathbf{x})$ is the true marginal density in $F - space$ and $\hat{P}_{\bar{F}}(\mathbf{x})$ is the estimated marginal density in the orthogonal complement $\bar{F} - space$ [3].

VI. System Architecture

The system architecture is made of three main modules: learning, face detection and face recognition. The first one is the learning process in which the system builds the eigenspace of the person with whom the robot is going to interact. Once this eigenspace is calculated the system is able to recognise the face of the person during the tracking process. For each captured image the system detects and extracts the faces, and projects them in the eigenspace of the person the robot is interacting with in order to know if it is interacting with the right person and where is the person in the image (see figure 5.

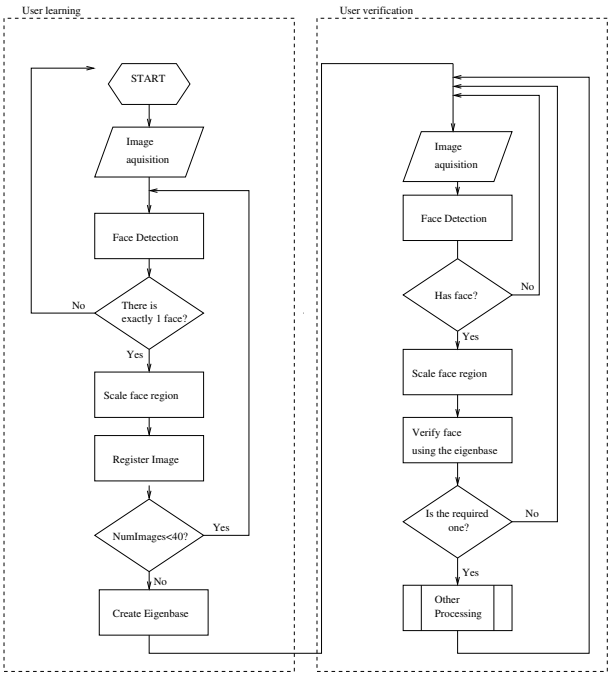

Fig. 5. System Architecture

A. Learning Process

The learning process starts with the acquisition of a face images sequence of the person the robot is going to interact with. The person should stay in front of the camera until face detector detects and extracts 40 face images.

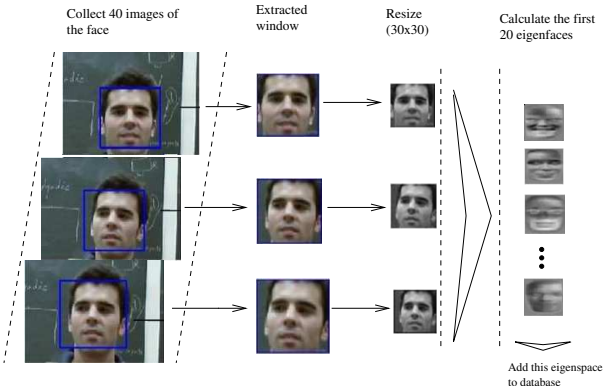

Fig. 6. Learning process

Every face image extracted is converted to grey level and scaled to 30×30 pixels. With this set of 40 grey level 30×30 face images the system is able to build the eigenspace of the person by calculating his first 20 eigenfaces (PCA). Fig. 6 illustrates the complete learning process of a person. It takes about 15 seconds in a 450 Mhz Pentium II processor.

B. Recognition Process

As in the learning process, the first stage of the recognition process is the detection and extraction of faces from the input image. Once this images were extracted they are scaled

Consider the linked list $faceslist$, with n nodes corresponding to the faces detected by the face detector, in a decreasing order of likelihood and consider $supthresh = 0.25$,$infthresh = 0.10$, $M = 100$:

1. If $facelist(1) \rightarrow likelihood > supthresh$ the robot finds the person it is interacting with.

2. If $n > 1$, $infthresh < facelist(1) \rightarrow likelihood < supthresh$ and $faceslist(1) \rightarrow likelihood > M \cdot faceslist(2) \rightarrow likelihood$ the robot finds the person it is interacting with.

3. Otherwise the person the robot is interacting with is not in the image acquired by the robot.

TABLE II
DECISION MECHANISM

to 30×30 pixels and projected in the eigenspace of the person the robot is interacting with. From the coefficients of projection the system is able to compute the probability of each detected person being the right one. The probability values are stored in a linked list in descendant order. Using a decision mechanism the system is able to know whether or not the robot is interacting with the right person and in the negative case the robot can recognise, among the people around, the person it should interact with.

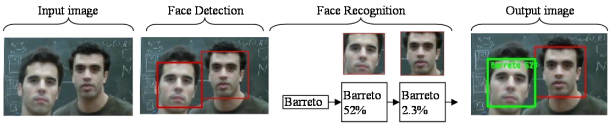

Fig. 7. Recognition process

In practice a very simple framework is used to produce an effective decision mechanism which is highly efficient (Tab. II). Facing deep changes in illumination conditions between the learning and recognition periods, the system behaves well mainly when more than one face is detected in the image since the second item of the decision mechanism can be applied.

C. Pre-Learnt User Recognition System

The system previously described was slightly changed in order to test it as a generic recognition system. The idea was to store the eigenspace of each person previously learnt by the system in the database. In this recognition system, every face detected in a frame is projected in the whole set of eigenspaces and then the probability values of being each known person is calculated and stored in a linked list. With an appropriate decision mechanism the system can identify the known faces among the detected ones. This kind of systems can be very useful not only to human-robot interaction applications, allowing the robot to interact with a set of known people, but also to vigilance and security applications.

VII. RESULTS

A 13 stage cascaded classifier was trained to detect frontal upright faces. Each stage was trained to eliminated 50% of the non-face patterns while falsely eliminating only 0.2% of the frontal face patterns. In the optimal case, we can expect a false alarm rate about $0.002^{13} = 8 \cdot 10^{-36}$ and a hit rate about $0.998^{13} = 0.97$ (see Fig. 3).

To train the detector, a set of face and nonface training images were used. The face training set consisted of over 4,000 hand labelled faces scaled and aligned to a base resolution of 24×24 pixels. The non-face subwindows used to train the detector come from over 6,000 images which were manually inspected and found to not contain any faces. Each classifier in the cascade was trained with the 4,000 training faces and 6,000 non-face sub-windows (also of size 24×24 pixels) using the Adaboost training procedure (Tab. I).

A. Selected Features

For the task of face detection, the initial rectangle features selected by AdaBoost are meaningful and easily interpreted. The first feature selected focus on the property that the region of the eyes is often darker than the region of the nose and cheeks. The second feature selected relies on the property that the eyes are darker than the bridge of the nose (see Fig. 8).

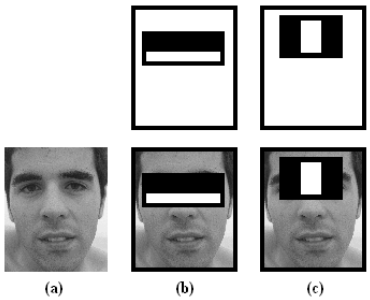

Fig. 8. The first and second features selected by AdaBoost. The two features are shown in the top row and then overlayed on a typical training face in the bottom row.

The final detector is scanned across the image at multiple scales and locations. Scaling is achieved by scaling the detector itself, rather than scaling the image. This process makes sense because the features can be evaluated at any scale with the same cost. The detector is also scanned across location. Subsequent locations are obtained by shifting the window some number of pixels Δ. Good results were obtained using a scale factor of 1.2 and $\Delta = 1.0$ pixels.

B. Recognition

As previously described the face recognition technique is based on eigenfaces. Good results were obtained for an eigenspace created with 20 eigenfaces. In the case of the pre-learnt user recognition system, experimental results show that the efficiency of the application decreases when the number of people in the database is bigger than 25. Above this number the discriminant ability of the PCA is not good enough to ensure the robustness of the system.

C. Speed of the Final Recognition System

The time the final recognition system takes to process one frame has two main components: detection time and recognition time. On a 450 Mhz Pentium II processor, the face detector can process a 320×240 pixel image in about 0.190 seconds and the recognition process of the faces returned by the face detector takes about 0.140. These times allow the system to process about 3 frames per second. In the pre-learnt user recognition system this value slightly decreases as the number of people in the database increases. In the worst case tested (25 people in the database) the system can process about 2 frames per second in the same processor.

The complete learning process of a person, previously described, takes about 15 seconds.

D. Hand Detection

Applying the technique used in the face detector we built a cascade of classifiers for hand detection being the hand a privileged vehicle for interaction with the robot.

The structure of this cascade is in Fig. 3 and once again has 13 stages each one with a maximum false alarm rate of 50% and a minimum hit rate of 99.8%. This cascade was trained with over 2,000 hand labelled upright hands scaled and aligned to a base resolution of 24×24 pixels. The non-face subwindows used to train the detector come from over 6,000 images which were manually inspected and found to not contain any hands.

The first rectangle feature selected by AdaBoost is meaningful. It focus on the property that the region in between the fingers is often darker than the region of the fingers (see Fig. 9).

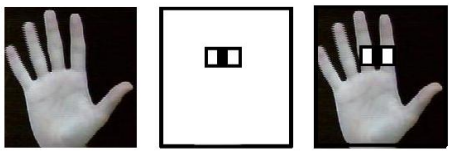

Fig. 9. The first feature selected by AdaBoost is shown in the middle and then overlayed on a typical training hand on the right.

In spite of the various hand degrees of freedom, allowing an infinite number of movements and deformations, the hand detector is quite robust on the detection of hands at various scales and with different backgrounds and illumination conditions. This hand detector can be very useful for example in the beginning of a human-robot interaction evolving gestures recognition since it gives the robot the information about the position of the hand whose gestures it should interpret. The hand detector processes 5.4 frames per second on a 450 Mhz Pentium II processor.

E. Experiments on Real-World Situations

The system was tested in some real-world situations and Fig. 10 presents a sequence of images captured by the robot's camera and processed by the real-time face recognition system.

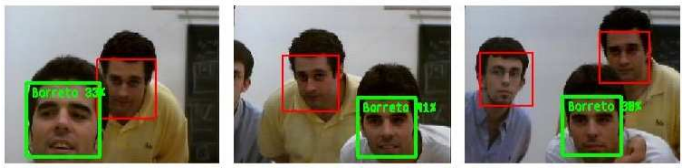

Fig. 10. Three frames from a Real-Time Face Recognition system output sequence.

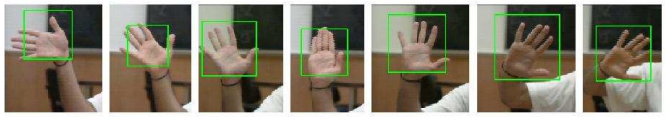

Fig. 11. A set of frames from a Hand Detection system output sequence.

Fig. 11 and 12 present hand detector output sequences. They show that some rotational movements and some degrees of deformation do not disturb the detector's performance.

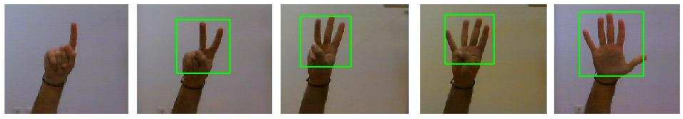

Fig. 12. The occlusion of less than 4 fingers does not affect the performance of the Hand Detection system.

VIII. Conclusions

It was presented an approach for real-time face recognition which can be very useful for human-robot interaction systems. In a human robot interaction environment this system starts with a very fast real-time learning process and then allows the robot to follow the person and to be sure it is always interacting with the right one under a wide range of conditions including: illumination, scale, pose, and camera variation. The face detection system works as a preprocessing stage to the face recognition system, which allows it to concentrate the face recognition task in a subwindow previously classified as face. This abruptly reduces the computation time. The introduction of a position predictive stage would also reduce the face search area driving to the creation of a robust automatic tracking and real-time recognition system.

This paper also presents a Pre-Learnt User Recognition System which works in almost real-time and that can be used by the robot to create a set of known people that can be recognised anytime. The robot has a certain number of people in the database and once a known face is found it can start following and interacting with it. Of course this system can also be used in security applications since it has the ability of searching a set of known people.

Finally, since the hand is a privileged vehicle of communication, it was also presented an approach for hand detection which minimises computation time while achieving high detection accuracy. Although not flexible enough to recognise a hand in every possible configuration, this mechanism can be

quite useful to initialise a hand tracker from one recognisable configuration.

REFERENCES

[1] P. Viola and M. Jones, "Rapid object detection using boosted cascade of simple features," in *Proceedings IEEE Conf. on Computer Vision and Pattern Recognition 2001*, 2001.

[2] R. Lienhart and J. Maydt, "An extended set of haar-like features for rapid object detection," in *IEEE ICIP 2002, Vol. 1, pp 900-903*, 2002.

[3] B. Moghaddam and A. Pentland, "Probabilist visual learning for object representation," *Technical Report 326, Media Laboratory, Massachusetts Institute of Technology*, 1995.

[4] A. K. Rainer Lienhart and V. Pisarevsky, "Empirical analysis of detection cascades of boosted classifiers for rapid object detection," *MRL Technical Report, Intel Labs*, 2002.

[5] M. Oren, C.Papageorgiou, P.Sinha, E.Osuna, and T.Poggio, "Pedestrian detection using wavelet templates," 1997.

[6] H. A. Rowley, S. Baluja, and T. Kanade, "Neural network-based face detection," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 20, no. 1, pp. 23–38, 1998.

[7] H. Schneiderman and T. Kanade, "A statistical method for 3D object detection applied to faces and cars," *In International Conference on Computer Vision*, 2000.

[8] K. Sung and T. Poggio, "Example-based learning for viewbased face detection," *In IEEE Patt. Anal. Mach. Intell.*, vol. 20, no. 1, pp. 39–51, 1998.

[9] M.-H. Yang, "Detecting faces images: A survey," *IEEE Transations on Pattern Analysis and Machine Inteligence*, vol. 24, no. 1, pp. 34–58, 2002.

[10] M. Turk and A. Pentland, "Face recognition using eigenfaces," *In Proc. of IEEE Conference on Computer Vision and Pattern Recognition*, pp. 586 – 591, 1991.

[11] A. Mohan, C. Papageorgiou, and T. Poggio, "Example-based object detection in images by components," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 23, no. 4, pp. 349–361, 2001.

[12] Y. Freund and R. E. Schapire, "A decision-theoretic generalization of on-line learning and an application to boosting," in *European Conference on Computational Learning Theory*, 1995, pp. 23–37.

[13] Y. Freund and R. E.Schapire, "Experiments with a new boosting algorithm," 1996.