Real-time Hand Gesture Detection and Recognition using Boosted Classifiers and Active Learning

Hardy Francke, Javier Ruiz-del-Solar and Rodrigo Verschae

Department of Electrical Engineering, Universidad de Chile
{hfrancke,jruizd,rverscha}@ing.uchile.cl

Abstract. In this article a robust and real-time hand gesture detection and recognition system for dynamic environments is proposed. The system is based on the use of boosted classifiers for the detection of hands and the recognition of gestures, together with the use of skin segmentation and hand tracking procedures. The main novelty of the proposed approach is the use of innovative training techniques - active learning and bootstrap -, which allow obtaining a much better performance than similar boosting-based systems, in terms of detection rate, number of false positives and processing time. In addition, the robustness of the system is increased due to the use of an adaptive skin model, a color-based hand tracking, and a multi-gesture classification tree. The system performance is validated in real video sequences.

Keywords: Hand gesture recognition, hand detection, skin segmentation, hand tracking, active learning, bootstrap, Adaboost, nested cascade classifiers.

1 Introduction

Hand gestures are extensively employed in human non-verbal communication. They allow to express orders (e.g. "stop", "come", "don't do that"), mood state (e.g. "victory" gesture), or to transmit some basic cardinal information (e.g. "one", "two"). In addition, in some special situations they can be the only way of communicating, as in the cases of deaf people (sign language) and police's traffic coordination in the absence of traffic lights. An overview about gesture recognition can be found in [18].

Thus, it seems convenient that human-robot interfaces incorporate hand gesture recognition capabilities. For instance, we would like to have the possibility of transmitting simple orders to personal robots using hand gestures. The recognition of hand gestures requires both hand's detection and gesture's recognition. Both tasks are very challenging, mainly due to the variability of the possible hand gestures (signs), and because hands are complex, deformable objects (a hand has more than 25 degrees of freedom, considering fingers, wrist and elbow joints) that are very difficult to detect in dynamic environments with cluttered backgrounds and variable illumination.

Several hand detection and hand gesture recognition systems have been proposed. Early systems usually require markers or colored gloves to make the recognition easier. Second generation methods use low-level features as color (skin detection) [4][5], shape [8] or depth information [2] for detecting the hands. However, those systems are not robust enough for dealing with dynamic environments; they

usually require uniform background, uniform illumination, a single person in the camera view [2], and/or a single, large and centered hand in the camera view [5]. Boosted classifiers allow the robust and fast detection of hands [3][6][7]. In addition, the same kind of classifiers can be employed for detecting static gestures [7] (dynamic gestures are normally analyzed using Hidden Markov Models [4]). 3D hand model-based approaches allow the accurate modeling of hand movement and shapes, but they are time-consuming and computationally expensive [6][7].

In this context, we are proposing a robust and real-time hand gesture detection and recognition system, for interacting with personal robots. We are especially interested in dynamic environments such as the ones defined in the *RoboCup @Home league* [21] (our *UChile HomeBreakers* team participates in this league [22]), with the following characteristics: variable illumination, cluttered backgrounds, real-time operation, large variability of hands' pose and scale, and limited number of gestures (they are used for giving the robot some basic information). In this first version of the system we have restricted ourselves to static gestures.

The system we have developed is based on the use of boosted classifiers for the detection of hands and the recognition of gestures, together with the use of skin segmentation and hand tracking procedures. The main novelty of the proposed approach is the use of innovative training techniques - active learning and bootstrap -, which allow obtaining a much better performance than similar boosting-based systems, in terms of detection rate, number of false positives and processing time. In addition, the robustness of the system is increased thanks to the use of an adaptive skin model, a color-based hand tracking, and a multi-gesture classification tree.

This paper is organized as follows. In section 2 some related work in hand gesture recognition and active learning is presented. In section 3 the proposed hand gesture detection and recognition system is described. In sections 4 and 5 the employed learning framework and training procedures are described. Results of the application of this system in real video sequences are presented and analyzed in section 6. Finally, some conclusions of this work are given in section 7.

2   Related Work

Boosted classifiers have been used for both hand detection and hand gesture detection. In [3] a hand detection system that can detect six different gestures is proposed. The system is based on the use of Viola&Jones' cascade of boosted classifiers [16]. The paper's main contributions are the addition of new rectangular features for the hand detection case, and the analysis of the gesture's separability using frequency spectrum analysis. The classifiers are trained and tested using still images (2,300 in total), which contains centered hands, with well-defined gestures. The performance of the classifiers in real videos is not analyzed. In [6] an extension of [3] is proposed, in which boosted classifiers are employed for hand detection, while gestures are recognized using scale-space derived features. The reported experiments were carried out in a dynamic environment, but using single, large and centered hands in the camera view. In [7] a real-time hand gesture recognition system is proposed, which is also based on the standard Viola&Jones system. New rectangular features for

the hand detection case are added. The recognition of gestures is obtained by using several single gesture detectors working in parallel. The final system was validated in a very controlled environment (white wall as background); therefore, its performance in dynamic environment is uncertain. In [9] a system for hand and gesture detection based on a boosted classifier tree is proposed. The system obtains very high detection results, however, the system is very time consuming (a tree classifier is much slower than a single cascade), and not applicable for interactive applications. Our main contribution over previous work are the use of a much powerful learning machine (nested cascade with boosted domain-partitioning classifiers), and the use of better training procedures, which increase the performance of the classifiers.

The performance of a statistical classifier depends strongly on how representative the training sets are. The common approach employed for constructing a training set for a learning machine is to use human labeling of training examples, which is a very time-consuming task. Very often, the amount of human power for the labeling process limits the performance of the final classifier. However, the construction of training sets can be carried out semi-automatically using active learning and the bootstrap procedure. This allows building larger training sets, and therefore to obtain better classifiers. Thus, the bootstrap procedure can be employed in the selection of negative samples [17]. The procedure requires that the human expert selects a large amount of images that do not contain object instances. During training, the bootstrap procedure automatically selects image areas (windows) that will be used as negative examples. In [11] the bootstrap procedure is extended for the particular case of the training of cascade classifiers. On the other hand, active learning is a procedure in which the system being built is used to lead the selection of the training examples. For instance, in [14] an interactive labeling system is used to select examples to be added to the training set. Initially, this system takes a rough classifier and later, interactively adds both, positive and negative examples. In the here-proposed approach both, bootstrap and active learning, are employed.
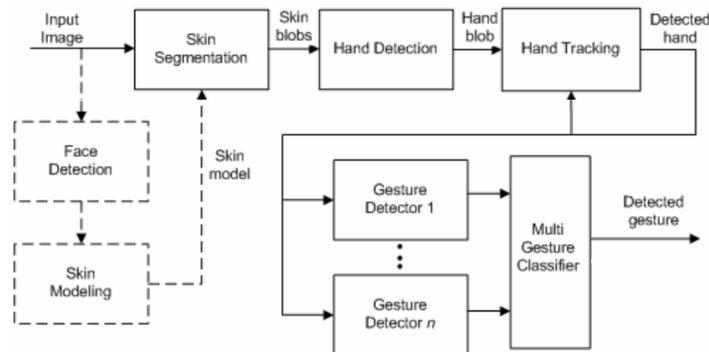
Figure 1: Proposed hand gesture detection and recognition system.

3 Real-time Hand Gesture Detection and Recognition System

3.1 System Overview

   The main modules of the proposed hand gesture detection and recognition system are shown in figure 1. The *Skin Segmentation* module allows obtaining skin blobs from the input image. The use of a very reliable face detector (*Face Detection* module) allows the online modeling of the skin, which makes possible to have an adaptive segmentation of the skin pixels. The *Hand Detection* and *Hand Tracking* modules deliver reliable hand detections to the gesture detectors. Hand detection is implemented using a boosted classifier, while hand tracking is implemented using the *mean shift* algorithm [1]. Afterwards, several specific *Gesture Detectors* are applied in parallel over the image's regions that contain the detected hands. These detectors are implemented using boosted classifiers [12]. Finally, a *Multi-Gesture Classifier* summarizes the detections of the single detectors. This multi-class classifier is implemented using a *J48 pruned tree (Weka's* [19] *version of the* C4.5 classifier*)*. In the next subsections these modules are described in detail.

3.2 Adaptive Skin Segmentation

   Adaptive skin segmentation is implemented using a procedure similar to the one described in [10]. The central idea is to use the skin color distribution in a perceived face to build a specific skin model. In other words, the skin model uses the context information from the person, given by its face, and the current illumination. With this we manage to have a robust skin detector, which can deal with variations in illumination or with differences in the specific skin's colors, in comparison to offline trained skin detectors. This approach requires having a reliable face detector. We employed a face detector that uses nested cascades of classifiers, trained with the Adaboost boosting algorithm, and domain-partitioning based classifiers. This detector is detailed described in [11].
   With the aim of making the model invariant to the illumination level to a large degree, the skin modeling is implemented using the RGB normalized color space:

$$I = R + G + B \; ; \; r = \frac{R}{I}; \; g = \frac{G}{I} \tag{1}$$

   After a new face is detected, a subset of the face pixels is selected for building the skin model (see figure 2). After pixels' selection and normalization, the *r*, *g* and *I* skin variables are modeled with Gaussian functions. The skin model parameters correspond to the variables' mean value and standard deviation: μ_r, σ_r, μ_g, σ_g, μ_I and σ_I. In order to lighten the computational burden, this modeling is carried out only once for every detected face (the first time that the face is detected). As long as there is not any major change in the illumination, there is no need to update the model. Having the skin model, the classification of the pixels is carried out as follows:

$$f(i,j) = \begin{cases} skin & if \quad |c - \mu_c| < \alpha_c \cdot \sigma_c, \quad c = r, g, I \\ non-skin & if \quad\quad\quad\quad\quad \sim \end{cases} \tag{2}$$

where i and j represent the coordinates of pixel being analyzed, and α_r, α_g and α_I are constants of adjustment of the classifier. For simplicity all these constants are made equal. In practice we have observed that this value needs to be adjusted depending on the brightness of the input image, increasing it when the brightness decreases, and vice versa.

After the skin pixels are detected, they are grouped together in skin blobs, according to their connectivity. In order to diminish the false positives from the skin detection, blobs that have an area below a certain threshold are discarded. Finally, all skin blobs are given to the next stage of the process except the ones containing faces.

3.3. Hand Detection and Tracking

In order to detect hands within the skin blobs, a hand detector is implemented using a cascade of boosted classifiers. Although this kind of classifiers allows obtaining very robust object detectors in the case of face or car objects, we could not build a reliable generic hand detector. This is mainly because: (i) hands are complex, highly deformable objects, (ii) hand possible poses (gestures) have a large variability, and (iii) our target is a fully dynamic environment with cluttered background. Therefore we decided to switch the problem to be solved, and to define that the first time that the hand should be detected, a specific gesture must be made, the fist gesture. Afterwards, that is, in the consecutive frames, the hand is not detected anymore but tracked. The learning framework employed for training the fist detector is described in section 4 and the specific structure of the detector in section 6.

The hand-tracking module is built using the mean shift algorithm [1]. The seeds of the tracking process are the detected hands (fist gestures). We use RGB color histograms as feature vectors (model) for mean shift, with each channel quantized to 32 levels (5 bits). The feature vector is weighted using an Epanechnikov kernel [1].

As already mentioned, once the tracking module is correctly following a hand, there is no need to continue applying the hand detector, i.e. the fist gesture detector, over the skin blobs. That means that the hand detector module is not longer used until the hand gets out of the input image, or until the mean shift algorithm loses track of the hand, case where the hand detector starts working again. At the end of this stage, one or several regions of interest (ROI) are obtained, each one indicating the location of a hand in the image.
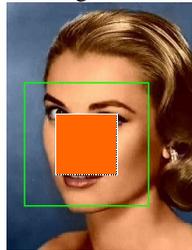
$$x_{0,orange} = x_{0,green} + 0.25 \cdot width_{green}$$
$$y_{0,orange} = y_{0,green} + 0.25 \cdot height_{green}$$
$$width_{orange} = 0.5 \cdot width_{green}$$
$$height_{orange} = 0.5 \cdot height_{green}$$

Figure 2: Left: The green (outer) square corresponds to the detected face. The orange (inner) square determines the pixels employed for building the skin model. Right: The orange square cropping formula.

3.4. Hand Gesture Recognition

In order to determine which gesture is being expressed, a set of single gesture detectors are applied in parallel over the ROIs delivered as output of the tracking module. Each single gesture detector is implemented using a cascade of boosted classifiers. The learning framework employed for building and training these classifiers is described in section 4. Currently we have implemented detectors for the following gestures: *first*, *palm*, *pointing*, and *five* (see Figure 3). The specific structure of each detector is given in section 6.

Due to noise or gesture ambiguity, it could happen than more than one gesture detector will give positive results in a ROI (more than one gesture is detected). For discriminating among these gestures, a multi-gesture classifier is applied. The used multi-class classifier is a *J48 pruned tree (Weka's* [19] *version of C4.5)*, built using the following four attributes that each single gesture detector delivers:

- *conf*: sum of the cascade confidence's values of windows where the gesture was detected (a gesture is detected at different scales and positions),
- *numWindows*: number of windows where the gesture was detected,
- *meanConf*: mean confidence value given by *conf/numWindows*, and
- *normConf*: normalized mean confidence value given by *meanConf/maxConf*, with *maxConf* the maximum possible confidence that a window could get.

| Fist | Palm | Pointing | Five |
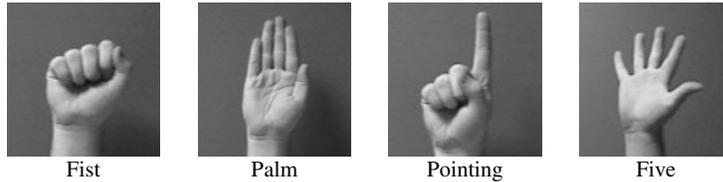
Figure 3: Hand gestures detected by the system.

4 Learning Framework

The learning framework used to train the hand detector and single gesture detectors is presented in the next subsections. An extensive description of this framework can be found in [11].

4.1. Learning using cascade of boosted classifiers

The key concepts used in this framework are nested cascades, boosting, and domain-partitioning classifiers. Cascade classifiers [16] consist of several layers (stages) of increasing complexity. Each layer can reject or let pass the inputs to the next layer, and in this way a fast processing speed together with high accuracy are obtained. Nested cascades [13] allow high classification accuracy and higher processing speed by reusing in each layer the confidence given by its predecessor. Adaboost [12] is employed to find highly accurate hypotheses (classification rules) by combining several weak hypotheses (classifiers). A nested cascade of boosted

classifiers is composed by several integrated (nested) layers, each one containing a boosted classifier. The cascade works as a single classifier that integrates the classifiers of every layer. Weak classifiers are linearly combined, obtaining a strong classifier. A nested cascade, composed of *M* layers, is defined as the union of M boosted classifiers H_C^k each one defined by:

$$H_C^k(x) = H_C^{k-1}(x) + \sum_{t=1}^{T_k} h_t^k(x) - b_k \qquad (3)$$

with $H_C^0(x) = 0$, h_t^k the weak classifiers, T_k the number of weak classifiers in layer *k*, and b_k a threshold (bias) value that defines the operation point of the strong classifier. At a layer *k,* processing an input *x*, the class assigned to *x* corresponds to the sign of $H_C^k(x)$. The output of H_C^k is a real value that corresponds to the confidence of the classifier and its computation makes use of the already evaluated confidence value of the previous layer of the cascade.

4.2 Design of the strong and weak classifiers

The weak classifiers are applied over features computed in every pattern to be processed. To each weak classifier a single feature is associated. Following [12], domain-partitioning weak hypotheses make their predictions based on a partitioning of the input domain *X* into disjoint blocks $X_1,...,X_n$, which cover all *X*, and for which $h(x)=h(x')$ for all *x, x'* $\in$ X_j. Thus, a weak classifier´s prediction depends only on which block, X_j, a given sample instance falls into. In our case the weak classifiers are applied over features, therefore each feature domain *F* is partitioned into disjoint blocks $F_1,...,F_n$, and a weak classifier *h* will have an output for each partition block of its associated feature *f*:

$$h(f(x)) = c_j \text{ s.t } f(x) \in F_j \qquad (4)$$

For each classifier, the value associated to each partition block (c_j), i.e. its output, is calculated so that it minimizes a bound of the training error and at the same time a loss function on the margin [12]. This value depends on the number of times that the corresponding feature, computed on the training samples (x_i), falls into this partition block (histograms), on the class of these samples (y_i) and their weight *D(i)*. For minimizing the training error and the loss function, c_j is set to [12]:

$$c_j = \frac{1}{2} \ln\left(\frac{W_{+1}^j + \varepsilon}{W_{-1}^j + \varepsilon} \right), \quad W_l^j = \sum_{i:f(x_i)\in F_j \wedge y_i = l} D(i) = \Pr\left[f(x_i) \in F_j \wedge y_i = l \right], \text{ where } l = \pm 1 \qquad (5)$$

where ε is a regularization parameter. The outputs, c_j, of each of the weak classifiers, obtained during training, are stored in a LUT to speed up its evaluation. The real Adaboost learning algorithm is employed to select the features and training the weak classifiers $h_t^k(x)$. For details on the cascade's training algorithm see [11].

4.3 Features

Two different kinds of features are used to build the weak classifiers, rectangular features (a kind of Haar-like wavelet) and mLBP (modified Local Binary Pattern). In

both cases the feature space is partitioned so that it can be used directly with the domain-partitioning classifier previously described. Rectangular features can be evaluated very quickly, independently of their size and position, using the integral image [16], while mLBP corresponds to a contrast invariant descriptor of the local structure of a given image neighborhood (see [15]).

5 Training procedures

The standard procedure to build *training sets* of objects and non-objects for training a statistical classifier requires that an expert (a human operator) obtains and annotates training examples. This procedure is usually very time-consuming; more importantly, it is very difficult to obtain representative examples. In the following, two procedures for solving these problems are presented.

5.1 Bootstrap Procedure

Every window of any size in any image that does not contain an object (e.g. a hand) is a valid non-object training example. Obviously, to include all possible non-object patterns in the training database is not an alternative. To define such a boundary, non-object patterns that look similar to the object should be selected. This is commonly solved using the bootstrap procedure [17], which corresponds to iteratively train the classifier, each time increasing the negative training set by adding examples of the negative class that were incorrectly classified by the already trained classifier. When training a cascade classifier, the bootstrap procedure can be applied in two different situations: before starting the training of a new layer (external bootstrap) and for re-training a layer that was just trained (internal bootstrap). It is important to use bootstrap in both situations [11]. The external bootstrap is applied just one time for each layer, before starting its training, while the internal bootstrap can be applied several times during the training of the layer. For details on the use of bootstrapping in the training of a cascade see [11].

5.2 Active Learning

As mentioned, the selection of representative positive training examples is costly and very time consuming, because a human operator needs to be involved. However, these training examples can be semi-automatically generated using active learning. Active learning is a procedure in which the system being built is used to lead the selection of the training examples.

In the present work we use active learning to assist the construction of representative positive training sets, i.e. training sets that capture the exact conditions of the final application. To generate training examples of a specific hand gesture detector, the procedure consists of asking a user to make this specific hand gesture for a given time. During this time the user hand is automatically tracked, and the bounding boxes (ROI) are automatically incorporated to the positive training sets of this gesture. If the hand is tracked for a couple of minutes, and the user maintains the

hand gesture while moving the hand, thousands of examples can be obtained with the desired variability (illumination, background, rotation, scale, occlusion, etc.). Thus, all windows classified as positive by the hand tracker are taken as positive training examples. This procedure can be repeated for several users. A human operator only has to verify that these windows were correctly detected, and to correct the alignment of the windows, when necessary. Later, all these windows are downscaled to the window size (24x24 or 24x42 pixels in our case) to be used during training.

In a second stage, active learning can also be employed for improving an already trained specific gesture detector. In this last case, the same procedure is employed (the user makes the hand gesture and the hand is tracked), but the already trained gesture detector is in charge of generating the training examples. Thus, every time that the gesture detector classifies a hand bounding box coming from the hand tracker as a non-object (the gesture is not detected), this bounding box is incorporated in the positive training set for this gesture.

6 Evaluation

In the present section an evaluation and analysis of the proposed system is presented. In this evaluation the performance of the system, as well as, its modules are analyzed. We also analyze the effect over the detector's performance of using Active learning during training. The detection results are presented in terms of Detection Rate (DR) versus Number of False Positives (FP), in the form of ROC curves. An analysis of the processing speed of the system is also presented.

The cascade classifiers were trained using three kinds of hand databases: (i) the IDIAP hand database [20], (ii) images obtained from the Internet, and (iii) images obtained using active learning and our hand gesture detection and recognition system. Table 1 and Table 2 summarize information about these training sets and the obtained cascade classifiers. For the other gesture's databases, the amount of data used to train the classifiers is similar.

On Table 1 and Table 2 we can also observe information about the structure of the obtained classifiers (number of layers and total number of weak classifiers). This information gives us an idea of the complexity of the detection problem, where large values indicate higher complexity and also larger processing times. These numbers are a result of the training procedure of the cascade [11] (they are not set a priori). As mentioned, we have selected a J48 pruned tree as multi-gesture's classifier. This classifier was trained using the training sets described in Table 3, using the Weka package, and 10-fold cross-validation. The obtained tree structure has 72 leaves and 143 tree nodes. In the validation dataset we obtained 90.8% of correct classifications.

To evaluate each single detector, a dataset consisting of 200 examples per class was used. This database contains images presenting a large degree of variability in the shape and size of the hands, the illumination conditions, and in the background. As a reference, this database contains more variability than the IDIAP database [20], and therefore is more difficult. The complete system was evaluated using a database that consists of 8,150 frames coming from 5 video sequences, where 4 different persons performed the 4 considered gestures. The sequences were captured by the same

camera used to perform the active learning, and emphasis was given to produce a cluttered background and varying illumination conditions.

To analyze how active learning improves the performance of the boosted classifiers, we studied two cases, a *fist* detector, and a *palm* detector. For each case we trained two classifiers, the first one using active learning and the second one without using it. The training of these detectors was done using the datasets presented in Table 1. The effect of using active learning in the performance of the detector is shown in Figure 4. To better show the effect of using active learning, the evaluation was performed by applying the detectors directly over the skin blobs in the input images that do not correspond to the face, i.e., not over the results of the hand-tracking module. As it can be noticed, the use of active learning during training largely improves the performance of the detectors, with up to a 90 % increase for operation points with low false positive rates. When using the tracking system, the number of false positives is reduced even more, so the complete system has much lower false positive rates than the ones observed here. Even though in the case of using active learning the obtained classifiers have a larger number of weak classifiers, the processing time is not much larger, because there is not a large increase on the number of weak classifier for the first layers of the cascade. As a consequence of this result, we choose to train all our gesture detectors using active learning.

An evaluation of the gesture detectors, trained using active learning, is shown in figure 5. In this case the results were obtained by applying the detectors directly over the detected skin blobs not corresponding to the face, not over the results of the hand-tracking module. The use of the hand tracking before applying the detectors reduces largely the number of false positives. The training was done using the datasets described in Table 2, and as in the previous experiment, the evaluation was done using a dataset consisting of 200 examples per class, which contains all gestures and a large degree of variability. As it can be observed, the *fist* gesture detector obtains a very high performance, achieving a detection rate of 99%, with just 2 false positives. The other detectors show a lower performance, having a higher number of false positives, which is reduced when the tracking module is used. The main reason for the large number of false positives is the large variability of the illumination conditions and background of the place where the detectors were tested. Figure 6 show some images from the test dataset, where it can be observed that it is an environment with several different light sources, and a lot of reflections, shadows, and highlights.

An evaluation of the complete system, that means the hand tracking and detection module, together with the gesture´s detection module and the gesture recognition´s module, is summarized in Table 4. The results are presented by means of a confusion matrix. The first thing that should be mention here is that the hand detection together with the tracking system did not produce any false positive out of the 8150 analyzed frames, i.e. the hands were detected in all cases. From Table 4 it can be observed that the gesture detection and recognition modules worked best on the *five* gesture, followed by the *pointing*, *fist* and *palm* gestures, in that order. The main problem is the confusion of the *fist* and *pointing* gestures, which is mainly due to the similarly of the gestures. In average the system correctly recognized the gestures in 70% of the cases. If the *pointing* and the *fist* gestures are considered as one gesture, the recognition rate goes up to 86%.

We also evaluated the processing time of the whole system. This evaluation was carried out in a PC powered with a Pentium 4 3.2GHz, 1GB RAM, running Windows XP and the system was implemented using the C language. The observed average processing time required for processing a 320x240 pixel's image, without considering the time required for image acquisition, was 89 milliseconds (see details in Table 5). With this, the system can run at about 11 fps for frames of 320x240 pixel size.

Table 1. Training sets for the *fist* and *palm* detectors. The D1 detectors are built using active learning, while the D2 detectors are built using standard hand databases.

Gesture	Size of training images (pixels)	Database	Training's set size	Validation's set size	Num. negative images	Num. layers	Num. weak classifiers
Fist (D1)	24x24	Active learning	1194	1186	46746	9	612
Fist (D2)	24x24	IDIAP [20]	795	606	47950	10	190
Palm (D1)	24x42	Active learning	526	497	45260	10	856
Palm (D2)	24x24	IDIAP [20]	597	441	36776	8	277

Table 2. Training sets and classifier structure for the definitive gesture's detectors.

Gesture	Size of training images (pixels)	Num. positive training images	Num. positive validation images	Num. negative (no-hand) images	Num layers	Total Num. detectors
Fist	24x24	1194	1186	46746	9	612
Palm	24x42	526	497	45260	10	856
Pointing	24x42	947	902	59364	12	339
Five	24x24	651	653	41859	9	356

Table 3: Training sets for the multi-gesture's classifier.

Gesture	Fist	Palm	Pointing	Five
Number of training examples	3838	3750	3743	3753
Number of training attributes	15352	15000	14972	15012

Table 4: Confusion matrix of the complete system.

Class\Predicted	Fist	Palm	Pointing	Five	Unknown	*Detection and recognition rates [%]*
Fist	1533	2	870	9	15	*63.1*
Palm	39	1196	10	659	15	*62.3*
Pointing	436	36	1503	27	86	*72.0*
Five	103	32	6	1446	127	*84.3*

Table 5: Average processing time of the main modules, in milliseconds
The size of frames is 320x240 pixels.

Skin detection	Face detection	Face tracking	Hand detection	Gesture recognition + Hand tracking
4.456	0.861	1.621	2.687	78.967

7 Conclusions

One of the ways humans communicate with each other is through gestures, in particular hand gestures. In this context, a framework for the detection of hands and the recognition of hand gestures was proposed, with the aim of using it to interact

with a service robot. The framework is based on cascade classifiers, a J48 tree classifier, an adaptive skin detector and a tracking system. The main module of the system corresponds to a nested cascade of boosted classifiers, which is designed to carry out fast detections with high DR and very low FPR. The system makes use of a face detector to initialize an adaptive skin detector. Then, a cascade classifier is used to initialize the tracking system by detecting the fist gesture. Afterwards, the hands are tracked using the mean shift algorithm. Afterwards, several independent detectors are applied within the tracked regions in order to detect individual gestures. The final recognition is done by a J48 classifier that allows to distinguishing between gestures.
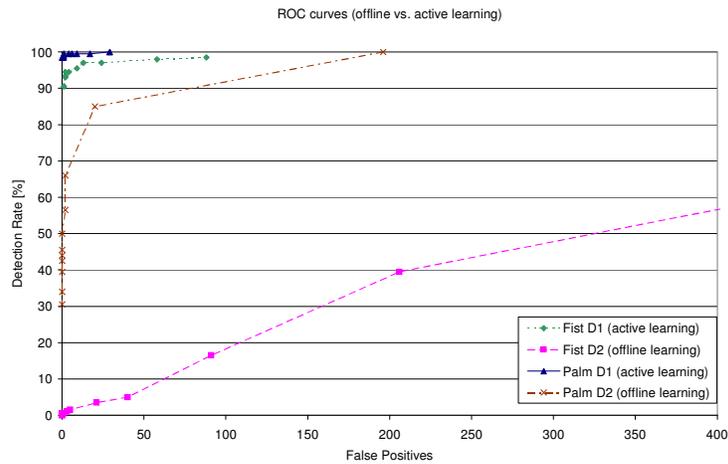
Figure 4: Fist and Palm detector ROC curves, using active learning (D1) and not using active learning (D2). In all cases the tracking system was not used.
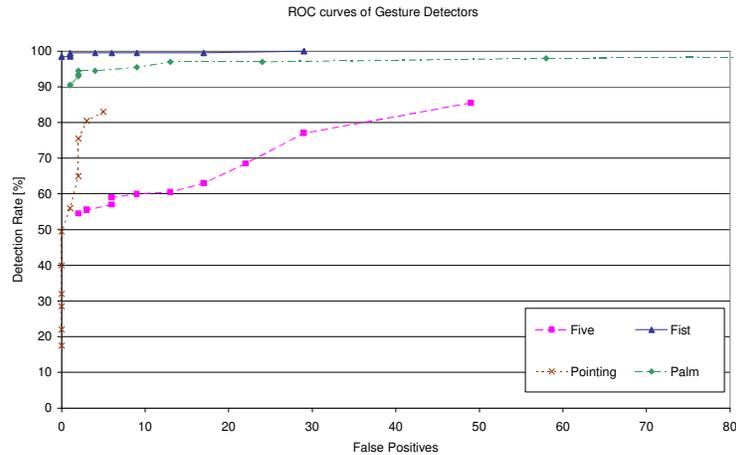
Figure 5: ROC curves of the gesture detectors (trained using active learning) applied without using the tracking system.

For training the cascade classifiers, active learning and the bootstrap procedure were used. The proposed active learning procedure allowed to largely increase the detection rates (e.g., from 17% up to 97% for the Palm gesture detector) maintaining a low false positive rate. As in our previous work [11], the bootstrap procedure [17] helped to obtain representative training sets when training a nested cascade classifier.

Out of the hand detectors, the best results were obtained for the *fist* detection (99% DR at 1 FP), probably because this gesture has the lower degree of variability. The worst results were obtained for the gesture *five* detector (85% DR at 50 FP), mainly because under this gesture the hand and the background are interlaced, which greatly difficult the detection process in cluttered backgrounds. In any case, it should be stressed that these results correspond to a worst case scenario, i.e. when no tracking is performed, and that when using the tracking the FPR is greatly reduced.

The system performs with a reasonable high performance in difficult environments (cluttered background, variable illumination, etc.). The tracking module has a detection rate over 99%, the detection module a 97% detection rate, and the gesture recognition rate is 70%. The main problem is the confusion of the *fist* with the *pointing* gesture and vice-versa. When these two gestures are considered as one, the global recognition rate goes up to 86%. We think that the recognition could be improved by using the history of the detection. The system presents a high processing speed (about 11 fps), and therefore it can be applied in dynamical environments in real time.

As future research we would like to extend our system for recognizing dynamic gestures and to improve the detection module by integrating the classifiers' cascades.

Acknowledgements

References

1. D. Comaniciu, V. Ramesh, and P. Meer, Kernel-Based Object Tracking, *IEEE Trans. on Pattern Anal. Machine Intell.*, vol 25, no. 5, (2003) pp. 564 – 575.
2. X. Liu and K. Hand gesture recognition using depth data, *Proc. 6th Int. Conf. on Automatic Face and Gesture Recognition*, (2004) pp. 529 – 534, Seoul, Korea.
3. M. Kolsch, M.Turk, Robust hand detection, *Proc. 6th Int. Conf. on Automatic Face and Gesture Recognition*, (2004) pp. 614 – 619, Seoul, Korea.
4. N. Dang Binh, E. Shuichi, T. Ejima, Real-Time Hand Tracking and Gesture Recognition System, *Proc. GVIP 05*, (2005) pp. 19-21 Cairo, Egypt.
5. C. Manresa, J. Varona, R. Mas, F. Perales, Hand Tracking and Gesture Recognition for Human-Computer Interaction, *Electronic letters on computer vision and image analysis*, Vol. 5, Nº. 3, (2005) pp. 96-104.
6. Y. Fang, K. Wang, J. Cheng, H. Lu, A Real-Time Hand Gesture Recognition Method, *Proc. 2007 IEEE Int. Conf. on Multimedia and Expo*, (2007) pp. 995-998

7. Q. Chen, N.D. Georganas, E.M. Petriu, Real-time Vision-based Hand Gesture Recognition Using Haar-like Features, *Proc. Instrumentation and Measurement Technology Conf. – IMTC 2007*, (2007)Warsaw, Poland

8. A. Angelopoulou, J. García-Rodriguez, A. Psarrou, Learning 2D Hand Shapes using the Topology Preserving model GNG, *Lecture Notes in Computer Science* 3951 (*Proc. ECCV 2006*), pp. 313-324

9. E.-J. Ong and R. Bowden, A boosted classifier tree for hand shape detection, *Proc. 6th Int. Conf. on Automatic Face and Gesture Recognition*, (2004) pp. 889 – 894, Seoul, Korea.

10. M. Wimmer, B. Radig, Adaptive Skin Color Classificator, Int. Journal on Graphics, Vision and Image Processing, Special Issue on Biometrics, vol 2 (2006) pp. 39-42.

11. R. Verschae, J. Ruiz-del-Solar, M. Correa, A Unified Learning Framework for object Detection and Classification using Nested Cascades of Boosted Classifiers, *Machine Vision and Applications* (in press).

12. R.E. Schapire, Y. Singer, Improved Boosting Algorithms using Confidence-rated Predictions, *Machine Learning*, 37(3): (1999) pp. 297-336.

13. B. Wu, H. Ai, C. Huang, S. Lao, Fast rotation invariant multi-view face detection based on real Adaboost, *Proc. 6th Int. Conf. on Automatic Face and Gesture Recognition*, (2004) pp. 79 – 84, Seoul, Korea.

14. Y. Abramson, Y. Freund, Active learning for visual object detection, *UCSD Technical Report* CS2006-0871, Nov. 19, 2006.

15. B. Fröba, A. Ernst, Face detection with the modified census transform, *Proc. 6th Int. Conf. on Automatic Face and Gesture Recognition*, (2004) pp. 91 – 96, Seoul, Korea.

16. P. Viola, M. Jones, Rapid object detection using a boosted cascade of simple features, *Proc. IEEE Conf. on Computer Vision and Pattern Recognition*, (2001) pp. 511 – 518.

17. K. Sung, T. Poggio, Example-Based Learning for Viewed-Based Human Face Deteccion, *IEEE Trans. Pattern Anal. Mach. Intell.*, Vol.20, No. 1, (1998) pp. 39-51.

18. The Gesture Recognition Home Page. Available on August 2007 in: http://www.cybernet.com/~ccohen/

19. Ian H. Witten and Eibe Frank (2005) "Data Mining: Practical machine learning tools and techniques", 2nd Edition, Morgan Kaufmann, San Francisco, 2005.

20. IDIAP hand gesture database. Available in Aug. 2007 in http://www.idiap.ch/resources/gestures/

21. RoboCup @Home Official website. Available in Aug. 2007 in http://www.robocupathome.org/

22. UChile RoboCup Teams official website. Available in Aug. 2007 in http://www.robocup.cl/
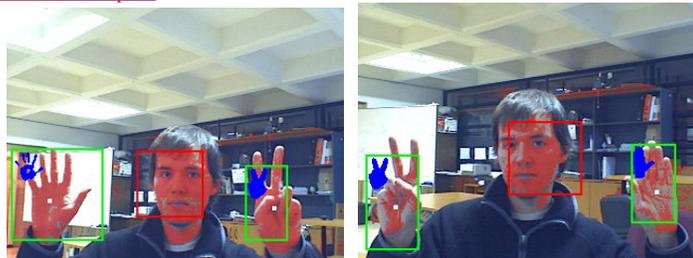
Figure 6: Example results of the system. The five, victory, and the victory gestures are detected and recognized. Notice the cluttered background, the highlights, and skin-like colors.