Technical Report UCH-DIE-VISION-2006-02

"A Unified Learning Framework for Face, Eyes and Gender Detection using Nested Cascades of Boosted Classifiers"

Rodrigo Verschae, Javier Ruiz-del-Solar, Mauricio Correa, Paul Vallejos

Computational Vision Group
Department of Electrical Engineering
Universidad de Chile

April 2006

A Unified Learning Framework for Face, Eyes and Gender Detection using Nested Cascades of Boosted Classifiers

Rodrigo Verschae, Javier Ruiz-del-Solar, Mauricio Correa, Paul Vallejos

Department of Electrical Engineering, Universidad de Chile, Santiago, CHILE
E-mail: rverscha@ing.uchile.cl, jruizd@ing.uchile.cl

Abstract

In this report is described a unified learning framework for face and eyes detection using nested cascades of boosted classifiers. The most interesting aspect of this unified learning framework is the possibility of building classification/detection systems with high accuracy (high detection rates with very low false positives), robustness (operation in dynamical environments), processing speed (real-time), and training speed (some hours in a standard PC). This framework should facilitate the task of building human-computer interfaces based on faces. Moreover, the same learning framework can be used for the construction of other object detection (e.g. hands, heads, pedestrians) and classification (e.g. gender, race) systems that can be also employed in the construction of visual human-computer interfaces. Using the proposed framework we were able to build state of the art face detection, eyes detection and gender classification systems. We have used these systems in the construction of different human-computer interfaces, namely: a system for multiple face detection and tracking in dynamic environments, a person detection and tracking system for AIBO robots, and tools for supporting the content-based image characterization on specific Web collections. All these systems and interfaces are described in this report.

Keywords: Human-Computer interaction through face analysis; boosting, nested cascade classifiers; face detection, eyes detection; gender classification.

1. Introduction

Visual human-computer interaction is of paramount importance in areas like robotics, medicine, telecommunications, entertainment, security and manufacturing. Among many other applications we can mention video conferencing, human-robot interaction, surveillance, computer games, computer interfaces, mobile devices communication, video summarizing, automatic television monitoring, image and video indexing and retrieval, customer profiling by observing attention, biometry, medical interfaces, and drivers monitoring.

In this context, face analysis plays an important role for building human-computer interfaces that allow humans to interact with computational systems in a natural way. Face information is by far, the most used visual cue employed by humans. There is evidence of specialized processing units for face analysis in our visual system. Faces play a vital role in our daily activities, with their varied repertoire of complex and often subtle functions. Faces allow us the localization and identification of other humans, and the interaction and visual communication with them. Therefore, if we want that humans can interact with machines with the same efficiency, diversity and complexity used in the human-human interaction, then face analysis should by extensively employed in the construction of such kind of interfaces.

Currently, computational face analysis (face recognition, face detection, eyes detection, face tracking, facial expression detection, etc.) is a very lively and expanding research field. The increasing interest in this field is mainly driven by some of the already mentioned applications, particularly the ones related with surveillance and security: access control to buildings and computational systems, identification for law enforcement, borders control, identification and

verification for credit cards and ATM, and passive recognition of criminals and terrorists in public places and buildings.

Face detection is a key step in almost any computational task related with the analysis of faces in digital images. Moreover, in many different situations face detection is the only way to detect persons in a given scene. Eyes detection can be considered as a specific kind of face detection, because it allows the detection of specific face parts and the enhancement of the face detections. Thus, eyes detection can help the process of detecting faces or it can be employed for face alignment after detection. Face detection and eyes detection are very challenging tasks, which should be performed robustly and efficiently, regardless variability in scale, location, orientation, pose, illumination and facial expressions, and considering possible object occlusions. In many applications the real-time requirement is added.

In this general context the aim of this work is to introduce a unified learning framework for face and eyes detection using nested cascades of boosted classifiers. The most interesting aspect of this unified learning framework is the possibility of building classification/detection systems with high: (i) accuracy (high detection rates with very low false positives), (ii) robustness (operation in dynamical environments), (iii) processing speed (real-time), and (iv) training speed (some hours in a standard PC). This framework should facilitate the task of building human-computer interfaces based on faces, with high requirements. Moreover, the same learning framework can be used for the construction of other object detection (e.g. hands, heads, pedestrians) and classification (e.g. gender, race) systems that can be also employed in the construction of visual human-computer interfaces.

Key concepts used in the proposed framework are boosting, nested cascade classifiers, and bootstrap training. Boosting is employed for finding (i) highly accurate hypotheses (classification rules) by combining several weak hypotheses (classifiers), each one having a moderate accuracy, and (ii) self-rated confidence values that estimate the reliability of each prediction (classification). In particular, we use the so-called *domain-partitioning weak hypotheses* learning paradigm [2] (called real Adaboost in [40]). Cascade classification uses several layers (stages) of classifiers of increasing complexity (each layer discards non-object patterns) for obtaining an optimal system in terms of classification accuracy and processing seed [36]. This is possible because of two reasons: (i) there is an important difference in the a priori probability of appearance of the classes, i.e. there are much more non-object than object patterns, and (ii) most of the non-objects patterns are quite different from the object patterns, therefore they can be easily discarded by the different layers. Nested cascade classification allows obtaining higher classification accuracy, by the integration of the different cascade layers. Each layer is considered as a component of its successor, i.e. the obtained confidence value of a given layer is employed as the first component of the boosted classifier of the next layer [40]. However, training issues are as important as learning issues when developing a complex learning machine. Special attention should be given to the selection of the training samples and to the training procedure. It is important to have training samples that correctly define the classification boundary. For defining such a boundary, non-object patterns that look similar to the objects should be selected. These selections can be done using the bootstrap procedure [32]: after a given instance of a classifier is trained, the current set of non-object examples can be enlarged with new non-object patterns that the current classification system wrongly classifies as objects.

Other aspects employed in the proposed framework for obtaining high-performance classification systems are: LUTs (Look-Up Tables) for a fast evaluation of the weak classifiers, simple rectangular features that can be evaluated very fast using the integral image [37], and LBP features [5] that are invariant against changing illumination.

Our most important improvements over previous work are mainly focused on the successful strategy for training the nested cascade of boosted classifiers, which allows us to obtain at the same time high classification accuracy (high detection rates with very low false positives), and fast training speed (about 15 hours in a standard PC). Our training strategy employs very extensively the bootstrap training paradigm, together with some heuristics that will be detailed described in this work.

Using the proposed framework we were able to build state of the art face detection, eyes detection and gender classification systems. We have used these systems in the construction of different human-computer interfaces, namely: (i) a system for multiple face detection and tracking in dynamic environments, (ii) a person detection and tracking system for AIBO robots, and (iii) tools

for supporting the content-based image characterization on specific Web collections. All these systems and interfaces are described in this report.

The report is structured as follows. In section 2 are outlined some related works to the topics of boosting, cascade leaning, face detection, and human-computer interfaces based on face analysis. In section 3 the proposed learning framework is presented, with special emphasis in the description of the training issues. In addition, face detection, eyes detection and gender classification systems, built using this framework are described. In section 4 is presented a comparative analysis of the critical components of the proposed learning framework, and a comparison of the trained face detection, eyes detection and gender classification systems with state of the art similar systems using standard image databases. In section 5 the implemented human-computer interfaces are described. Finally, some conclusions and projections of this work are given in section 6.

2. Related Work

Several approaches have been proposed for the computational detection of faces in digital images. A very comprehensive review can be found in [8][44]. Main approaches can be classified as: (i) feature-based, which uses low-level analysis (e.g. color, edges, textures), feature analysis or active shape models, and (ii) image-based, which employs linear subspace methods, neural networks or statistical analysis. Image-based approaches have shown a much better performance than feature based [8][44].

Starting with the seminal works or Rowley [24] and Sung & Poggio [32], successful image-based proposed approaches include the use of neural networks [6], SNoW classifiers [23], Bayesian classifiers [43], SVM [21], and boosted cascades [37][19]. This last approach is based on a new learning paradigm introduced by Viola and Jones in [36]. This paradigm consists on using a cascade of boosted classifiers to obtain a very fast system that is capable at the same time of achieving high detection rates.

Adaboost [28], a boosting algorithm, has shown to have very good results in different kinds of classifications problems and because of its simplicity, high performance, and fast speed, it has been started to be widely used. In the face detection context, Adaboost has been used not only for cascade systems (see for example [30]). Also other kinds of boosting procedures have been developed and implemented for detections system, one example is [19], in which they introduce kullback-leiber boosting, which is a boosting algorithm based on the kullback-leiber divergence.

The boosted cascade paradigm has been widely studied and extended in the last few years, being [40][5][18][19][29] some of the most important works in this area, with some of them achieving the best reported results on face detection. The key idea to obtain a fast detection is that the complexity of the classifiers (i.e. number of features) increases when advancing in the cascade. In this way for windows that are easy to classify, not much processing is performed and they are discarded in a short time. Cascade classifiers have been also used for the detection of other kind of objects, like cars [29], hands [15], pedestrians [39], traffic signs [29] and rotated faces [29] [38].

An improved version of [36] proposed by the same authors in [37] outperforms previous systems in terms of processing speed, by keeping a good recognition rate. This system uses simple, rectangular features (a kind of Haar wavelets), a cascade of filters that discard non-face windows, the integral image for fast computation of these filters, and asymmetrical real Adaboost as a boosting strategy for the training of the detectors. It is worth to mention that when using real Adaboost, a confidence value of each detected face is calculated. That means that the detection results are not binary (yes/no) but real values.

The system developed by Wu et al. [40] uses the idea described in [28] concerning the use of domain-partitioning weak classifiers, which gives a very important improvement in the representation power of the weak classifiers, compared to the ones used in [37], by keeping a simple representation that at the same time increases the accuracy of the classifier, reduces the processing time and training time. Another interesting idea introduced by [40] is the use of *nested* cascades. A similar idea is presented in [42]. Nested cascades are cascades that use the confidence output of a layer in the next layer of the cascade, which produces more compact and accurate cascades.

In [5] Fröba et al. introduced the use on a different kind of feature based on LBP (local binary patterns) which are robust to extreme illumination condition and also introduces the use of what we call internal bootstrapping during the training of the cascade (see section 3 for details). They use a cascade consisting of 4 layers, the 3 first layers are boosted classifiers, and the last one is a SNoW [23] classifier that uses LBP features. In [18] a system robust to occlusions based on cascade classifiers in presented. Different kinds of occlusions are defined. In case that a possible occluded face is detected by any layer of the cascade, a parallel cascade, specially designed for that kind of occlusion is started from that layer to classify that windows. In [29] several cascades are trained to detect not only faces, but also cars, traffic signals, cups, etc. These cascade are based on what they call a feature-centric approach, in opposite to the window-centric approach, like the one used by [40][5][18][19]. The feature-centric approach re-uses feature evaluation from adjacent windows. The classifier used in [29] is a boosted semi-naïve Bayes classifier and the features are based on a wavelet representation of the image.

Eyes detection systems can be built using the same learning concepts employed for building face detection systems. In the case of the learning paradigms here analyzed, that means, using cascade of boosted classifiers. Two recently proposed systems that use this paradigm are described in [20] and [3]. They use Adaboost and Gentleboost, as the learning algorithm, respectively. In another related work, in [13], an eye detector that is trained in the same way that the Viola & Jones face detector [37] is used in a face recognition system. No information is given about the accuracy of the resulting eye detection system.

In the context of gender classification, works related to our approach are [41] and [31]. [41] uses a LUT-based Adaboost system for the gender classification that uses rectangular features. Prior to the classification the faces are aligned. This is done through a face alignment method called SDAM that is a kind AAM (Active Appearance Model). This alignment locates 3 landmark points (eyes and mouth). They achieve classification rates of 88% on images downloaded form Internet. In [31] a gender classification system that is also trained for race (asian/non-asian) classification is presented. That system is a boosted classifier that uses rectangular features. They obtain classification rates of about 79% on a set of faces obtained from Internet that were manually annotated prior to the classification.

Most of the described concepts related with the design, construction and training of boosted cascade were considered in our unified learning framework. The proposed learning framework corresponds to a nested cascade of boosted classifiers, and it is based on the seminal ideas proposed by Viola and Jones in [37], with the later improvements introduced in [5] and [40]. In this framework we employ a nested cascade of filters [40], weak partitioning real Adaboost as a boosting strategy for the training of the filters [40], LUTs for a fast evaluation of the weak classifiers [40], rectangular and LBP features [5], and the integral image for a fast computation of the rectangular features [37].

It is important to mention that using weak partitioning real Adaboost, a confidence value for each detected object (face, eyes or gender) is calculated. This allows the integration of the different cascade layers (stages), because each layer can be considered as a component of its successor. More specifically, the obtained confidence value of a given layer is employed as the first component of the boosted classifier of the next layer.

Our most important improvements over previous work are mainly focused on the adequate training of the nested cascade of boosted classifiers, which allows us to obtain at the same time high classification accuracy (high TPR = True Positive Rate with very low FPR=False Positives Rate), and fast training speed (about 15 hours in a standard PC). Our training strategy employs very extensively the bootstrap training paradigm, together with some heuristics that will be detailed described in next section.

3. Proposed Learning Framework for Face, Eyes and Gender Detection

In this section we describes the proposed learning framework (section 3.1), with especial emphasis in the description of the training procedures (section 3.2). In addition, we describes a face detection system (section 3.3), an eyes detection system (section 3.4) and a gender classification system (section 3.5) built using this framework

3.1. An Integrated Nested Cascade Boosting Learning Framework

A nested cascade of boosted classifiers is composed by several integrated (nested) layers, every one containing a boosted classifier. The whole cascade works as a single classifier that integrates the classifiers of every layer. In the following paragraphs our realization of this concept will be explained.

A nested cascade C, composed of M layers, is defined as the union of M boosted (or strong) classifiers H_C^k:

$$C = \bigcup_{k=1}^{M} \{H_C^k\} \tag{1}$$

Each H_C^k is defined by:

$$H_C^k(x) = H_C^{k-1}(x) + \sum_{t=1}^{T_k} h_t^k(x) - b_k \tag{2}$$

with

$$H_C^0(x) = 0 \tag{3}$$

and $h_t^k(x)$ the so-called weak classifiers, T_k the number of weak classifiers in layer k, and b_k a threshold value.

It should be noted that a given classifier corresponds to the nesting (combination) of the previous classifiers. The output of H_C^k is a real value that corresponds to the confidence of the classifier ($conf_C^k$). The computation of H_C^k makes use of the already evaluated $conf_C^{k-1}$, the confidence value of the previous layer of the cascade (see figure 1).

Due to the nested configuration of C, its output is given by:

$$O_C(x) = \begin{cases} \text{sign}(H_C^l(x)) & H_C^k(x) \geq 0, k = 1,...,l \wedge H_C^{l+1}(x) < 0 \\ \text{sign}(H_C^1(x)) & H_C^1(x) < 0 \end{cases} \tag{4}$$

with a confidence value of a positive detection given by:

$$conf_C(x) = H_C^l(x) \ni H_C^k(x) \geq 0, k = 1,...,l \wedge H_C^{l+1}(x) < 0 \tag{5}$$

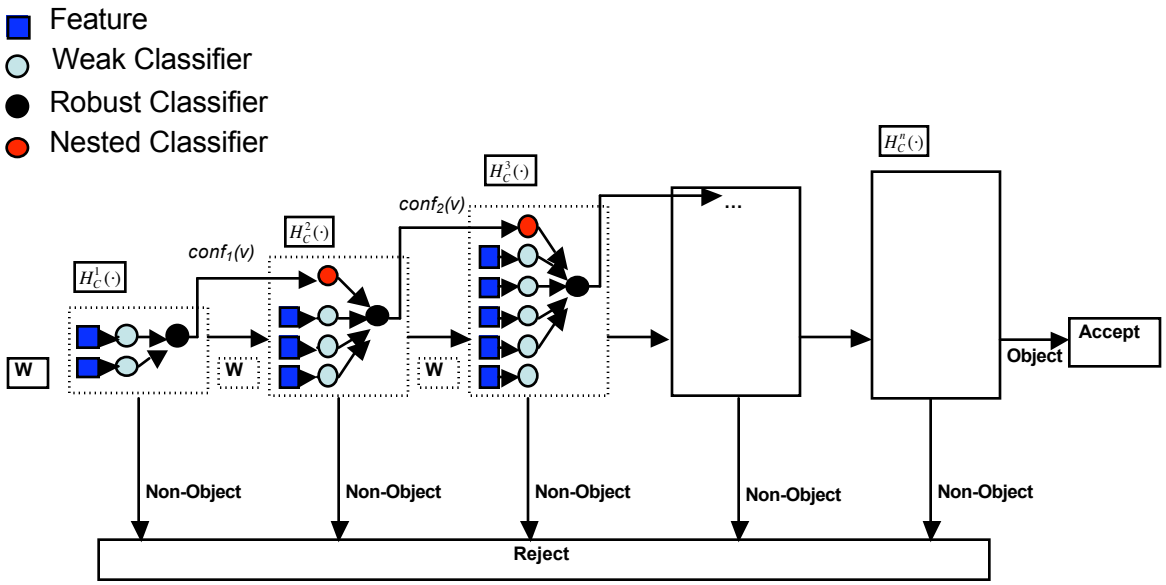


Figure 1. Block diagram of an Adaboost nested cascade.

The weak classifiers are applied over features computed in every pattern to be processed. Each weak classifier has associated a single feature. The features we employ are rectangular Haar-like features [37] and LBP features [5]. Rectangular features are simple features that can be evaluated very quickly, independently of their size, using the integral image [37]. LBPs (Local Binary Patterns) are also known as texture numbers or census transform, and are invariant to changing illumination. We use their modified version described in [5].

The weak classifiers are designed after the *domain-partitioning weak hypotheses* paradigm [28]. Under this paradigm the weak classifiers make their predictions based on a partitioning of the domain X. Each classifier has a value associated with a partition of this domain. The domain is partitioned into disjoint blocks $X_1,...,X_n$, which cover all of X, and for which $h(x) = h(x')$ for all $x,x' \in X_j$. Thus, the weak classifiers 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:

$$h(f(x)) = c_j \ni f(x) \in F_j \tag{6}$$

For each classifier, the value associated to each partition block (c_j), i.e. its output, is calculated for minimizing a bound of the training error [28]. This value depends on the number of times that the corresponding feature, computed on the training samples (x_i), fall into this partition block (histograms), on the class of these samples (y_i), and on their importance $D(i)$. This value is given by [28]:

$$c_j = \frac{1}{2}\ln\left(\frac{W_{+1}^j + \varepsilon}{W_{-1}^j + \varepsilon}\right) \tag{7}$$

with

$$W_l^j = \sum_{i:f(x_i)\in F_j \wedge y_i = l} D(i) = \Pr\left[f(x_i) \in F_j \wedge y_i = l\right], \text{ where } l = \pm 1 \tag{8}$$

and ε a regularization parameter [28].

The real Adaboost learning algorithm [28] is employed for selecting the features and training the weak classifiers $h_t^k(x)$. The implemented real Adaboost learning algorithm takes into account the nested configuration of the cascade (a given layer of the cascade can not be trained without taking

into account the result of previous trained layers, see (2)-(4)), and the asymmetrical distribution of the two classes. The pseudo code of this algorithm is shown in figures 2, 3 and 4.

It is important to notice that the threshold values b_k are chosen for minimizing the number of boosted classifiers of the layer. In each iteration (through the *ValidateClassifier* function, see figure 4), several values of $b_k > 0$ are tested for fulfilling the classification requirements (TPR and FPR), procedure that allows the selection of the minimum number of possible weak classifiers needed for each layer. This aspect is very important because the classification speed of the cascade depends directly on the number of classifiers per layer.

The output of the weak classifiers (c_j), obtained during training, is stored in a LUT for speeding up its evaluation (see figure 5). Thus, a weak classifier will be defined by:

$$h(x) = h_{LUT}[x] = LUT[index_{bin}(f(x)]$$

with $index_{bin}$ a function that returns the index (bin) associated to the feature value $f(x)$ in the LUT.

In the case of using rectangular features, their domain can be partitioned in bins of equal size [40] or of variable size. We have experimented with variable size bins, which require more processing time during training, but the results we have obtained are similar to the ones obtained when using bins of equal size. Therefore we use bins of equal size. In the case of LBP features, the partition is already defined by the feature itself (each feature is represented by a binary number of 9 bits).

PT : Positive Training Set

PV : Positive Validation Set

NT : Negative Training Set

NV : Negative Validation Set

$tpr(H,P)$: true positive rate of a classifier H evaluated on the set P

$tprMin$: minimum allowed true positive rate for a layer

$fpr(H,N)$: false positive rate of ta classifier H evaluated on the set N

$fprMax$: maximum allowed false positive rate for a layer

Figure 2. Notation for the real Adaboost learning algorithm.

RealAdaboostTraining$(H_C^k, H_C^{k-1}, PT, NT, PV, NV, fprMax, tprMin)$**{**

Given $(x_1, y_1),...,(x_m, y_m)$ were $x_i \in X = NT \cup PT, y_i \in Y = \{-1,+1\}$

set $D_1(i) \leftarrow \left\{ \begin{matrix} \frac{1}{|NT|} & x \in NT \\ \frac{1}{|PT|} & x \in PT \end{matrix} \right\}, i = 1,...,m$

$D_1(i) \leftarrow D_1(i)\exp(-y_i H_C^{k-1}(x_i)), i = 1,...,m$

normalize $D_1(i)$ to a p.d.f., $i = 1,...,m$

$H_C^k(x) \leftarrow H_C^{k-1}(x)$

$t \leftarrow 1$

while$(\neg\, ValidateClassifier(H_C^k, k, PV, NV, fprMax, trpMin)$**{**

// train all weak classifiers :

for each feature $f_p \in F, h_p \in H,\ p = 1,...,P$ **do** {

$$W_l^j = \sum_{i: f_p(x_i) \in F_j \wedge y_i = l} D_t(i), \text{ where } l = \pm 1,\ j = 1,...,J$$

$$h_p(j) = \frac{1}{2}\ln\left(\frac{W_{+1}^j + \varepsilon}{W_{-1}^j + \varepsilon}\right), j = 1,...,J$$

calculate the normalization factor : $Z_p = 2\sum_j \sqrt{W_{+1}^j W_{-1}^j}$

}

select h_t, which is the one that minimizes Z, i.e,

$$Z_t = \min_{h_p \in H} Z$$

$$h_t = \arg\min_{h_p \in H} Z$$

$H_C^k(x) \leftarrow H_C^k(x) + h_t(x)$

$D_{t+1}(i) \leftarrow D_t(i)\exp(-y_i h_t(x_i)), i = 1,...,m$

normalize $D_{t+1}(i)$ to a p.d.f., $i = 1,...,m$

$t \leftarrow t + 1$

}

}

Figure 3. Real Adaboost training algorithm for a layer k of a nested cascade.

ValidateClassifier$(H_C^k, k, PV, NV, frpMax, trpMin)$\{
  for b_l in $B, l = 1,...,L$ **do**
    $\overline{H} \leftarrow H_C^k - b_l$
    if $(fpr(\overline{H}, NV) \le \prod_{l=1}^{k} frpMax \wedge tpr(\overline{H}, PV) \ge \prod_{l=1}^{k} trpMin)$\{
      $H_C^k \leftarrow \overline{H}$
      return TRUE
    \}
  return FALSE
\}

Figure 4. Classifier validation procedure.

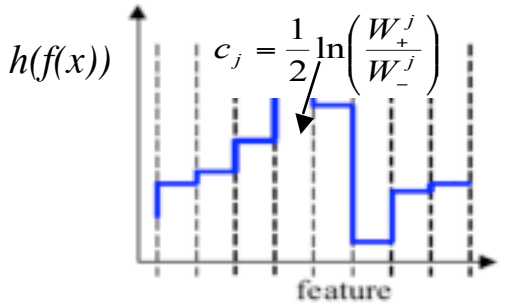


Figure 5. LUT associated to a weak classifier *h(x)*.

3.2. Training Procedure

When developing a complex learning machine, as for example a nested cascade of boosted classifiers, special attention should be given to the training process. It is not only important the adequate selection of the training examples, they should be statistically significant, but also their distribution between the different classes, and the way in which they are "shown" to the learning machine. In a nested cascade of boosted classifiers, which corresponds to a learning machine with several interrelated layers that are trained at different moments, it is not obvious which the best training strategy is. That is, which examples to present to a specific part of the machine in a given moment? Which criterion to use for stopping the training of a given part of the machine? How is related this criterion with the required performance of the machine (TPR and FPR)?

Moreover, detection problems require discriminative analysis between the object and the rest of the world. That produces a very high asymmetry in the classes from the point of view of a two-class classification problem. The problem is how to select the non-object training examples. From the several millions examples (the rest of the world), one should select the ones that correctly define the classification boundary (positive class versus negative class). How to select the adequate negative examples when training a specific part of a nested cascade of boosted classifiers is not obvious.

These questions can be answered using the bootstrap procedure [32], and some heuristic that will be described in this section.

If we take as a case study a face detection system, every non-face window of any size in any image not containing faces is a valid training example. Obviously, to include all possible non-face patterns in the training database is not an alternative. As mentioned, important is to have training samples that correctly define the classification boundary. For defining such a boundary, non-face patterns that look similar to faces should be selected. These patterns can be selected using the bootstrap procedure: after a given instance of a classifier is trained, the current set of non-face examples can be enlarged with new non-face patterns that the current classification system wrongly classifies as faces.

However, when training a nested cascade consisting of several layers it is not obvious where, how oft and how many times to make the bootstrap. After our experience it is important to use bootstrap during the training process of a given layer, we call it *intra-layer* or *internal bootstrap*, and before starting the training of a new layer, we call it *inter-layer* or *external bootstrap*. The inter-layer bootstrap is applied just one time, before starting the training of a layer, while the intra-layer bootstrap can be applied several times during the training of the layer. The bootstrap procedure in both cases is the same (see figure 6).

The training procedure of the whole nested cascade is described in figure 7 and 8. The nested cascade is trained until the target overall FPR is achieved. The training of each cascade layer includes the use of 1 external bootstrap (applied before training), and B internal bootstraps (applied after training). Every time an internal bootstrap is applied, the layer under training is rebuilt (reset).

```
Bootstrap(CASCADE, SIZE, NEG_IMGS_SET){
    N ← {φ}
    while(|N| < SIZE){
        x ← Sample(NEG_IMGS_SET) // extracts a random example window of the set of images
        if (O_CASCADE(x)) ≥ 0) // the negative sample is classified as positive
            N ← N ∪ {x}
    }
    return N
}
```

Figure 6. Bootstrap procedure.

So far we have explained the implementation of cascade classifiers and their training, but we have not explained in detail why they are attractive for solving object detection problems. The main reason for using them is the possibility of having an optimal system in terms of classification accuracy and processing seed. This is possible because of two reasons: (1) there is an important difference in the a priori probability of appearance of the classes, i.e. there are much more non-object than object windows, and (2) most of the non-objects windows are quite different from the object windows, therefore they can be easily discarded, i.e. classified as non-objects.

Taking these two points into consideration, it can be notice that the average processing time of the windows is defined by the processing time of the non-object windows. Therefore the idea is to process most non-object windows as fast as possible, and to process carefully the object windows and the non-object, object-like windows. That is, to discard as soon as possible non-object windows in each layer of the cascade. This can be achieved by forcing each layer of the cascade to have the lowest number of features, while achieving a high true positive rate and a low false positive rate. As already mentioned, we attain this by adjusting the bias of each layer of the cascade (figure 4, *classifier validation* procedure).

An important point in the training of a complex nested cascade is the time it takes to accomplish it. For example in a system like the one presented in [37], the training time can takes from weeks to months in a single Pentium 4 computer. One of the major improvements of our system is the short training time. In the case of the face detection here presented (see section 3.3), the training time takes about 15 hours in a standard PC. This is possible thanks to: (1) the use of a nested cascade, (2) the implementation of domain-partitioning weak classifiers implemented using LUTs, (3) the use of internal bootstraps, (4) the use of features that can be evaluated very fast, rectangular (integral image) and LBP, (5) the used of the bias for forcing the lowest number of features in each stage, and (6) the use of some heuristics. The most important heuristic is that, at each iteration of the Adaboost algorithm, not all possible features are considered for selection, but only a randomly selected subset of them. This feature sampling reduces considerable the training time without affecting very much the performance of the final system. These aspects are analyzed in detail in section 4.1.

NestedCascadeTraining(){

$C \leftarrow \{\phi\}$ //Nested cascade C, initially empty

$H_C^0 \leftarrow 0$

$k \leftarrow 0$ //Cascade layers counter

do{

$k \leftarrow k + 1$

$NT_k \leftarrow Bootstrap(C, InitSizeNT, NegIMTrainSet)$

$NV_k \leftarrow Bootstrap(C, SizeNV, NegIMValSet)$

//Train layer k of C

for $b = 1...B$ **do** // Number of internal bootstrapps

- RealAdaboostTraining($H_C^k, H_C^{k-1}, PT, NT_k, PV, NV_k, frpMaxL, trpMinL$)
- $\overline{C} \leftarrow C \bigcup H_C^k$
- $BNT_k \leftarrow Bootstrap(\overline{C}, (FinalSizeNT - InitSizeNT)/B, NegIMTrainSet)$
- $NT_k \leftarrow NT_k \cup BNT_k$

$C \leftarrow \overline{C}$

}**while**($fpr(C, NV_k) > fprMaxC$)

return C

}

Figure 7. Nested Cascade training procedure using intra- and inter-layer bootstrap.

PT : Positive Training Set

PV : Positive Validation Set

NT_k : Negative Training Set at layer k

NV_k : Negative Validation Set at layer k

BNT_k : Bootstrapped Negative Training Set at layer k

$tprMinL$: minimum allowed true positive rate of a Layer

$fpr(C,N)$: false positive rate of the cascade evaluated on the set N

$fprMaxL$: maximum allowed false positive rate of a layer

$fprMaxC$: target overall false positive rate of the cascade

$NegIMValSet$: set of images containing non-positives patterns (to be used in the validation set)

$NegIMTrainSet$: set of images containing non-positives patterns (to be used in the training set)

$SizeNV$: size of the bootstrapped validation set of negative windows

$InitSizeNT$: inicial size of the training set of negative windows

$FinalSizeNT$: final size of the training set of negative windows

Figure 8. Notation for nested cascade training.

3.3. Face Detection

3.3.1. General Organization

The block diagram of the face detector is presented in figure 9. First, for detecting faces at different scales a multiresolution analysis is performed by scaling the input image by a factor of 1.2

(*Multiresolution Analysis* module). This scaling is performed until images of about 24x24 pixels are obtained. Afterward, windows of 24x24 pixels are extracted in the *Window Extraction* module, for each of these scaled versions of the input image. The extracted windows can be then pre-processed for obtaining invariance against changing illumination, like variance normalization [36] or histogram equalization [24]. In our system thanks to the use of LBP features and the partitioning of the rectangular features, we do not perform any kind of preprocessing.

Afterwards the pre-processed windows are analyzed by the already described nested cascade of boosted classifiers (*Cascade Classification Module*). Each of the cascade layers feed backs the confidence value of the classification to the *Window Extraction* module. This information is employed for deciding if the image, in the corresponding window position, needs to be further processed in a lower scale (resolution) or not. When a window is classified as non-face by a given layer, non-further processing is done. In the opposite case, the window goes further in the cascade.

After all selected windows are processed and classified as faces or non-faces, in the *Overlapping Detection Processing* module the face windows are analyzed and fused (normally a face will be detected at different scales) for determining the size and position of the final detections. In this module the confidence values associated to the detections are used for fusing detections. If the number of overlapped face windows in a given position is larger than a given threshold th_{num}, then they are considered as a true detection and fused. Also, if the *detection volume* [6] of the overlapped face windows in a given position is larger than a threshold th_{vol}, then they are considered as a true detection and fused. The detection volume is defined as the sum of all confidences values corresponding to a set of the overlapped windows corresponding to a face. The fusion procedure is described in [34].

3.3.2. Algorithms Flavors

Different flavors of the face detection system are implemented. They are briefly described in the following paragraphs.

- Full search versus Speed search

The difference between *Full Search* and *Speed Search* is that in the full search case all possible image windows are considered (as shown in figure 9), while in the speed search a multi-grid approach is employed. The speed search is based in the procedure described in [4]. As already mentioned, images are analyzed in different resolutions by scaling them by a factor of 1.2, until images of about 24x24 pixels are obtained. For the speed search approach, each level of the pyramid is analyzed in three stages, in a coarse to fine manner. In a first stage (*s1*) only image positions on a sparse grid with grid step *step1*=6 are used for extracting windows. In this way only about 2.8% of all possible positions are evaluated. Each grid position with a score value (the confidence value feed back by the filters) below a threshold *ths1* is a starting point for a local refinement of the search. In the second stage (*s2*) a finer structured grid around each starting point of the coarse grid is evaluated, using a grid step *step2=3*. Then, each grid position with a score value below a threshold *ths2* is evaluated in the third stage (*s3*) using a 3x3 neighborhood. The speed up of the process is controlled by the threshold's values *ths1* and *ths2*. Higher threshold's values means higher processing speed (less fine scales are evaluated), but lower detection rates (many true faces can be lost). Because we are using a nested cascade, we have to set different values of *ths1* and *ths2* for each of the layers. This was done using a multi-objective genetic algorithm (see [35] for details).

- All faces versus Most relevant faces

There are some face detection applications where it is required to detect all possible faces (e.g. surveillance), while in others is required to detect just one (e.g. passport processing) or the most relevant ones (e.g. video conference). For the second case we have implement a variant of our algorithm that detects just the most relevant faces in a given image or video frame. The most relevant faces procedure consists on detecting all faces in a given image, but to filter out the faces

that have a confidence value (CV) much lower than the CV of the face with the largest CV. In our implementation much lower means 20 times lower.

Summarizing, we have four variants of our face detection algorithm: *Full-All* (full search, all faces), *Full-Most* (full search, most relevant faces), *Speed-All* (speed search, all faces) and *Speed-Most* (speed search, most relevant faces).

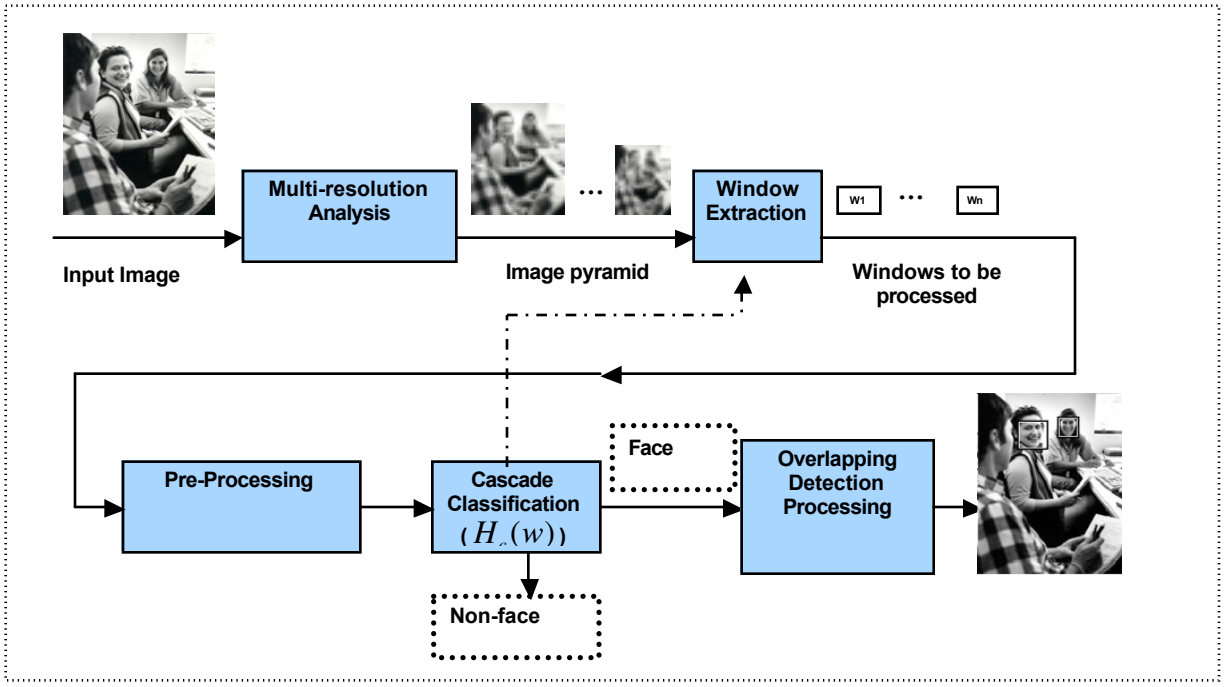


Figure 9. Block diagram of a face detection system.

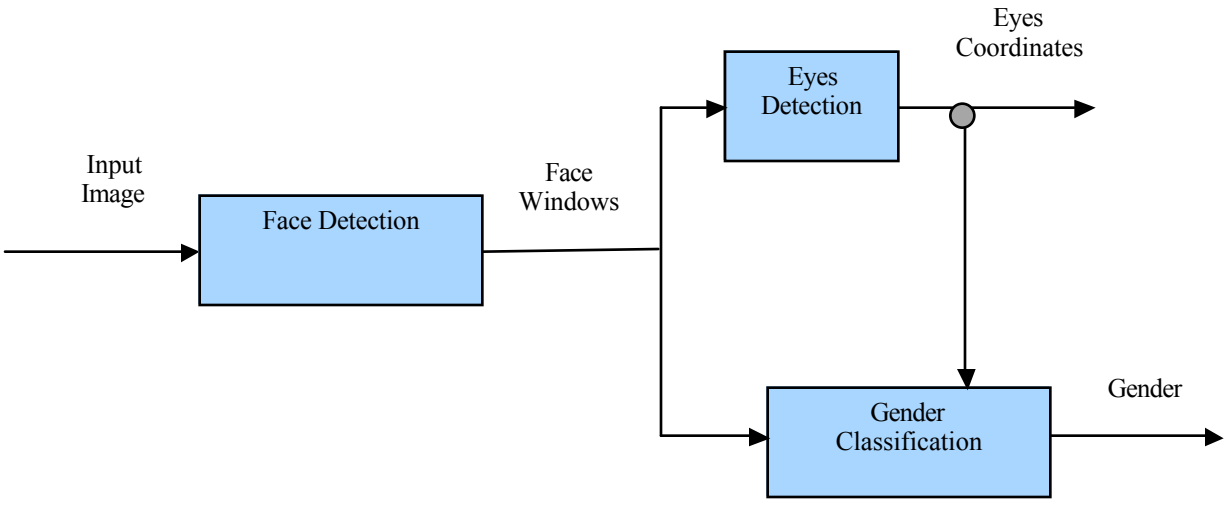


Figure 10. Block diagram of the integrated face detection, eyes detection and gender classifications systems.

3.4. Eyes Detection

The eye detector follows the same ideas that the face detector does, i.e. it has the processing modules (see figure 9). The only difference is that the search for the eyes is performed in the upper part of the face area, i.e. the *Window Extraction* module extracts windows from within the face area (see figure 10). A left eye detector is used to process the left upper part of the detected face, and a right eye detector is used in the same way in the right upper part of the face. Only one eye detector has to be trained, in this case we have trained the left eye detector; the other is a mirrored (flopped) version of the one that was trained. Because there are at most two eyes per face, for the eye detector, the *Overlapping Detection Processing* returns at most one left eye and at most one right eye.

The left eye detector we have trained corresponds to a 2-layer cascade, it works on windows of 24x24 pixels, and its weak classifiers are based on rectangular features. The training set used for the eye detector is as follows: the eye examples where extracted from faces contained in the Yale, YaleB and PIE databases, and the same non-face images where used to collect the initial training set of non-eyes and for the bootstrapping of the non-eyes. For generating the eyes for the training and validation sets, randomly selected zooms, translations and rotations were applied to left eyes and mirrored right eyes, obtaining about 16,500 eyes for the training set and 7,650 eyes for the validation set. Three internal bootstrapping steps are performed for each layer, collecting 400 examples each time. During the external bootstrapping an initial set of 2400 examples is collected for the training of each layer. The maximum FPR per layer was 20%, and the minimum TPR per layer was set to 0.9975.

Because the eye detector will be applied to a restricted, reduced image area (upper part of a face), only one flavor for the eye detector is needed, which is equivalent to the full-search used for the face detector. Instead of using all-faces or most-relevant-faces flavors, the eye detector returns at most one left and one right eye.

3.5. Gender Classification

Following the general block diagram of figure 9, for the gender classifier the only required module needed is the *Cascade Classification* module. The gender classifier works on low resolution face images, that is on windows of 24x24 pixels. It can work on pure faces or on faces aligned by using information given by the eye detector (see figure 10).

The gender classifier consists of just one layer classifier that uses LBP features. The initial training set consists on 3,140 annotated faces of adult persons, 1,526 male faces and 1,614 female faces. After the training using the initial set, three (internal) bootstrapping steps were performed in which in total 150 manually annotated male face images were added to the database. The bootstrapping was performed only for male faces, because for the female faces the classification rate was quite good (initially it was close to 100%). After that, the face detector was applied to the original images of the training gender set, and the correctly detected faces where added to the training set and the gender classification system was trained again.

We also trained the gender classifier on larger windows sizes and with different zooms (to add also parts of the hair of the head), but the obtained results were similar or worst to the ones obtained with 24x24 pixels window sizes.

4. Experimental Methodology and Results

In this section we will present a comparative analysis of the critical components of the proposed learning framework. We will concentrate mainly in the analysis of the different learning procedures and heuristics. Afterwards we will compare the trained detection and classification systems (face, eyes and gender) with state of the art similar systems using standard image databases.

4.1. Analysis of training parameters

We have performed a comparison of the different improvements and variations proposed for designing and training cascades of boosted classifiers. In this analysis we compared the following elements: (1) the use of normal cascades versus nested-cascades, (2) the application of what we call intra-layer bootstrapping (internal-bootstrapping), (3) the use of rectangular and/or LBP features, and (4) the effect of different parameters that have to be selected for running the training procedure of the cascade, such as the maximum FPR in each layer and the sampling of features.

The whole analysis presented in this section was carried out on the face detection problem, namely, using face detectors built using the proposed learning framework. The obtained results should be valid for other classification or detection systems (eyes, gender, race, etc.) to be built using the same framework.

For carrying out this analysis more than 9,000 images, obtained from different sources such as Internet and familiar photograph albums, were employed. No single image employed for testing (see section 4.2) was employed in this analysis. The following image datasets were built:

- PT (Positive Training set): about 5,000 training face examples obtained from several thousand images.
- PV (Positive Validation set): the mirror (flop) of all faces from the PV.
- NIT (Negative Images Training set): 3500 images containing non-faces and used for the bootstrap during training.
- NIV (Negative Images Validation set): 1500 images containing non-faces and used for the bootstrap during validation. These images are different from the ones in NIT.
- NT_k (Negative Training set for layer k): the negative non-face examples employed for the training of layer k, and obtained using bootstrap from NIT. Due to the use of internal and external bootstrap, this set change in each layer and in each iteration.
- NV_k (Negative validation set for layer k): the negative non-face examples employed for the validation of a layer, and obtained using bootstrap from NIV. Due to the use of internal and external bootstrap, this set change in each layer and in each iteration.

The presented analysis was performed using ROCs. In these ROCs, each operation point was obtained by evaluating cascade instances with different number of layers, using the validation sets PV and NV_k. In more simple words, the parameter to be changed for obtaining the ROCs is the number of layers of the cascades.

For all the following experiments the minimum TPR per layer was set to 0.999, which gives for n layers, a cascade TPR of 0.999^n (with 10 layers 0.99045).

4.1.1 Nested versus Non-nested Cascades

In the figure 11(a) is shown the effect of using a nested cascade versus a non-nested cascade in terms of classification accuracy, while in figure 11(b) is shown the number of features obtained for each layer of the cascades. In these graphs it can be seen that in the first layers of the cascade the non-nested cascade has larger FPR per layer than the nested one. Moreover, it can be noticed that for the first layers of the cascade, the non-nested cascades needs more than two times the number of features required by the nested cascade. Because of the training time is directly related with the number of features, the training time of a nested cascade is about a half of the training time of a non-nested cascade.

4.1.2 Internal Bootstrap

As already explained in section 3.2, we propose to use both internal (intra-layer) and external (inter-layer) bootstrap. After several experiments we found out that it is better to repeat the internal bootstrap several times, 3 times in the case of our training datasets. However, we wanted to quantify the real effect of this internal bootstrap in the performance of the whole cascade. Figure 11(c) shows the effect of performing internal bootstrap. Clearly the effect for the first layers (the ones with larger FPR) is quite important, and the use of internal bootstrap reduces the FPR to the half, and the number of selected features is almost the same.

4.1.3 Feature Sampling during Training

For reducing the training time, in iteration of the Adaboost feature (classifier) selection process not all features are considered, but a subset of the possible features [34][1]. In this way it is possible to reduce the training time. We have tested the training algorithm using 100% and 20% (randomly sampled) of the features in each Adaboost iteration. In figure 11(d) are shown the obtained results. As it can be seen using more features does not clearly give better results, but seems to give a more stable curve. Therefore, we use only the 20% of the features for reducing the training time.

4.1.4 LBP versus Rectangular Features

We have tested the use of rectangular and LBP features. We make three different experiments: training using only LBP features, training using only rectangular features, and training using both kind of features, rectangular ones for the first two layers and LBP features for the subsequent layers. As it can be notice in figure 11(e), the use of only LBP features has a much lower performance than using only rectangular features, but when using both kind of features, the performance is not greatly affected compared with the situation when only rectangular features are employed.

But, why it is interesting to use LBP features and not just rectangular features? First, LBP features are invariant to difficult illumination conditions, and they have shown to have a better performance than rectangular features in images with extreme illumination [5]. Second, the number of LBP possible features to be selected is much smaller than the number of rectangular features; therefore, the training is much faster when using LBP features[1]. Taking that into consideration and the fact that when using both kinds of features a similar performance is obtained compared with the situation when using just rectangular features, in our final detection and classification systems we use rectangular features in the first two cascade layers and LBP features in the subsequent ones.

4.1.5 Selection of Maximum FPR per Layer

The selected maximum FPR per layer has an important effect in the performance of the final system. The effect is even counter intuitive. As it can be noticed in the figure 11(f), the use of larger maximum FPRs per layer reduces the performance of the system instead of increasing it. We think that this is mainly because of two reasons: (1) the internal bootstrap has a much greater effect when more difficult examples are bootstrapped, which happens when the maximum FPR is smaller, and (2) large maximum FPRs can produce the selection of very few classifiers in one layer, which have negative effects in the following ones. Thus, in some of the experiments done by Schapire et al. in [28], for boosted classifiers of sizes from 1 to 5-10, the error increases when adding more classifiers, however after adding at least 10 classifiers or more, the error rates starts to diminished very quickly. In our case, when the maximum FPR is large (0.5 in figure 11(f)), only 4 weak classifiers are needed in the first layer, which is a low value for a boosted classifier. On the other hand, when the maximum FPR is low (0.35 in figure 11(f)), the number of selected weak classifiers in the first layer is 7.

[1] When using processing windows of 24x24 pixels and 3x3 pixel neighborhoods for evaluating the LBP features, there are about 135,000 possible rectangular features and only 484 LBP features. Therefore, the training (mainly selection of features and classifiers) using only LBP features is about 257 times faster.

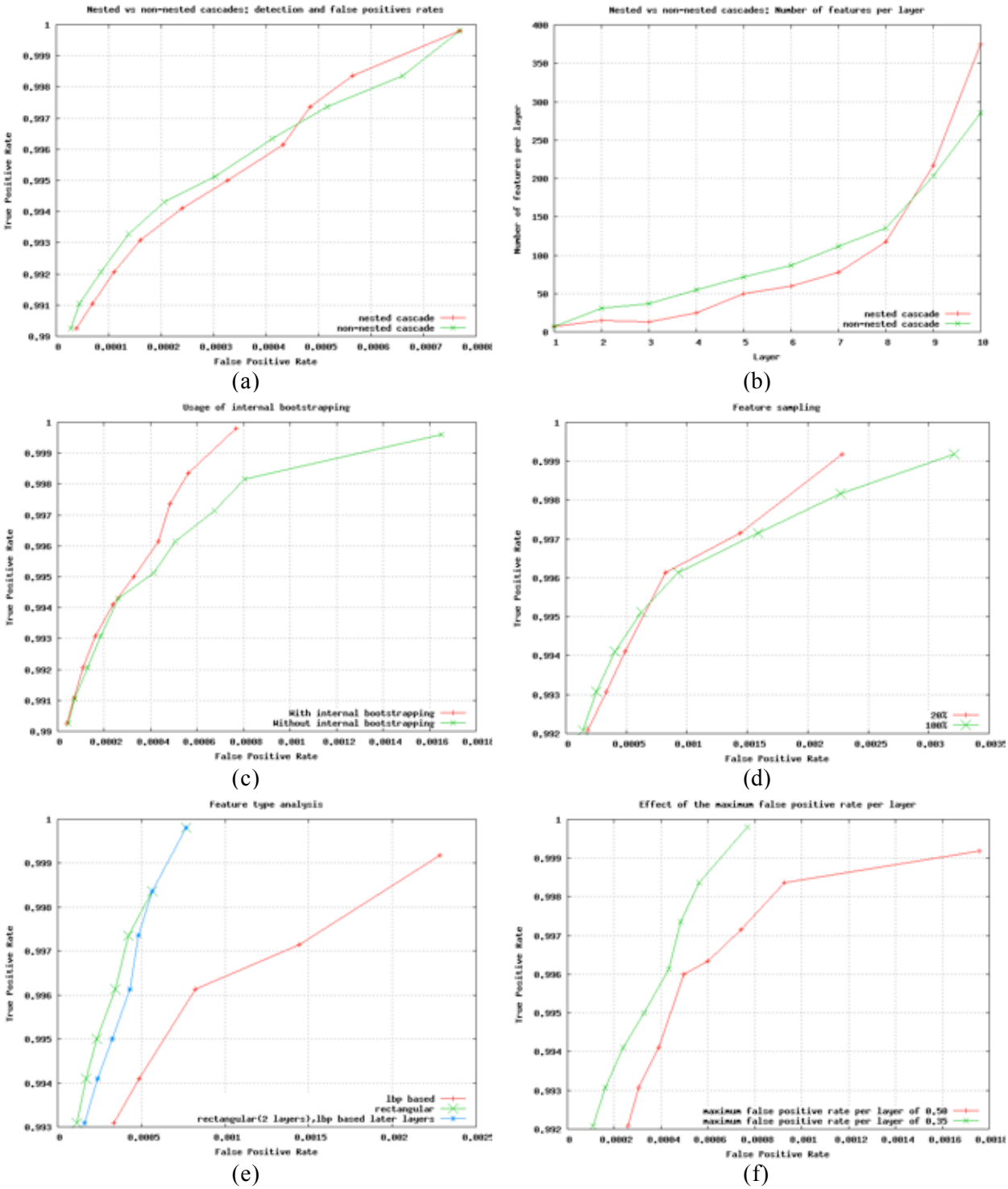


Figure 11. Analysis of training parameters. (a) Evaluation of nested and non-nested cascade on the validation set. (b) Number of features needed at each stage of the nested and non-nested cascades. (c) Effect of using internal bootstrap. (d) Effect of feature sampling during training. (e) Effect of selecting different features types. (f) Effect of selecting different maximum FPR per layer.

Summarizing, our final face detection system uses a nested cascade composed by domain-partitioning weak classifiers implemented using LUTs. The employed features are rectangular features for the first two layers and LBP-based for the subsequent layers. The cascade was trained

using a maximum FPR per layer of 0.20, a minimum TPR per layer of 0.999, and 3 internal bootstraps for the training of each layer. The initial negative training set for each layer had 2400 examples and in each internal bootstrap steps 400 more examples were added, obtaining a total of 3600 negative examples for the final training of each layer.

The final training time of the whole nested cascade was about 15 hours in a 1.8 GHz Pentium 4, with 1,280 MB running Debian Linux. This is quite good compared to [37] which takes weeks to be trained.

4.2 Evaluation of the Proposed Classifiers

4.2.1. Experimental Datasets

For testing our face detection, eyes detection and gender classification systems we employed three standard face databases (BioID, FERET and CMU-MIT), and a new face database (UCHFACE). No single image from theses databases was used for the training of our systems.

The BioID Face Database [45] consists of 1,521 gray level images with a resolution of 384x286 pixel. Each one shows the frontal view of a face of one out of 23 different test persons. During the recording special emphasis has been laid on "real world" conditions, therefore the test set features a large variety of illumination, background and face size. The database contains manually set eye positions. We added the labeling of gender information (available in [47]).

The FERET database [22] was assembled to support testing and evaluation of face recognition algorithms using standardized tests and procedures. The final corpus consists of 14,051 eight-bit grayscale images of human heads with views ranging from frontal to left and right profiles. For compatibility with our previous study about face recognition algorithms [26], we selected 1,016 images containing frontal faces with annotated eyes (254 persons, 4 images for each person) for testing our detection systems. The employed FERET subset and the gender labeling are available in [47].

The CMU-MIT database [24] consists of 130 grayscale images containing 507 faces. It was originally created for evaluating algorithms for detecting frontal views of human faces. Due to its low resolution and low quality (some images are very noisy), it cannot be employed for detecting eyes but for gender classification. It is important to notice that in some publications people has used different subsets of this dataset, therefore comparison is not always straight forward. In this case we use all 130 images and we considered all 507 faces. Notice that some of the annotated faces are face drawings.

The UCHFACE database was especially created for evaluating gender classification and eye detection algorithms in images obtained under uncontrolled conditions. We also use it for testing our face detection algorithms. It consists of 142 grayscale images containing 343 frontal and semi-frontal faces. Face, eyes and gender information is annotated in all these images, which were obtained from Internet. This database is available for future studies in [47].

4.2.2 Face detection evaluation

- Face detection in Single Face Images: BioID database

In figure 12(a) are shown the ROC curves of the different face detector flavors on the BioID database. Some selected points of these ROCs are shown in table 1 for comparing these results with the best results reported in the literature in this database. In [5] Fröba reported “while achieving comparable results on the CMU sets, we reach the best published results on the BioID database”. In table 1 it can be seen that our results are much better than to the ones reported by Fröba, especially in the lower parts of the ROCs (low FPR). Other authors reported detection rates of 91.8% [12] and 92.8% [14], but they do not indicate the FPR. In any case these numbers are lower than ours.

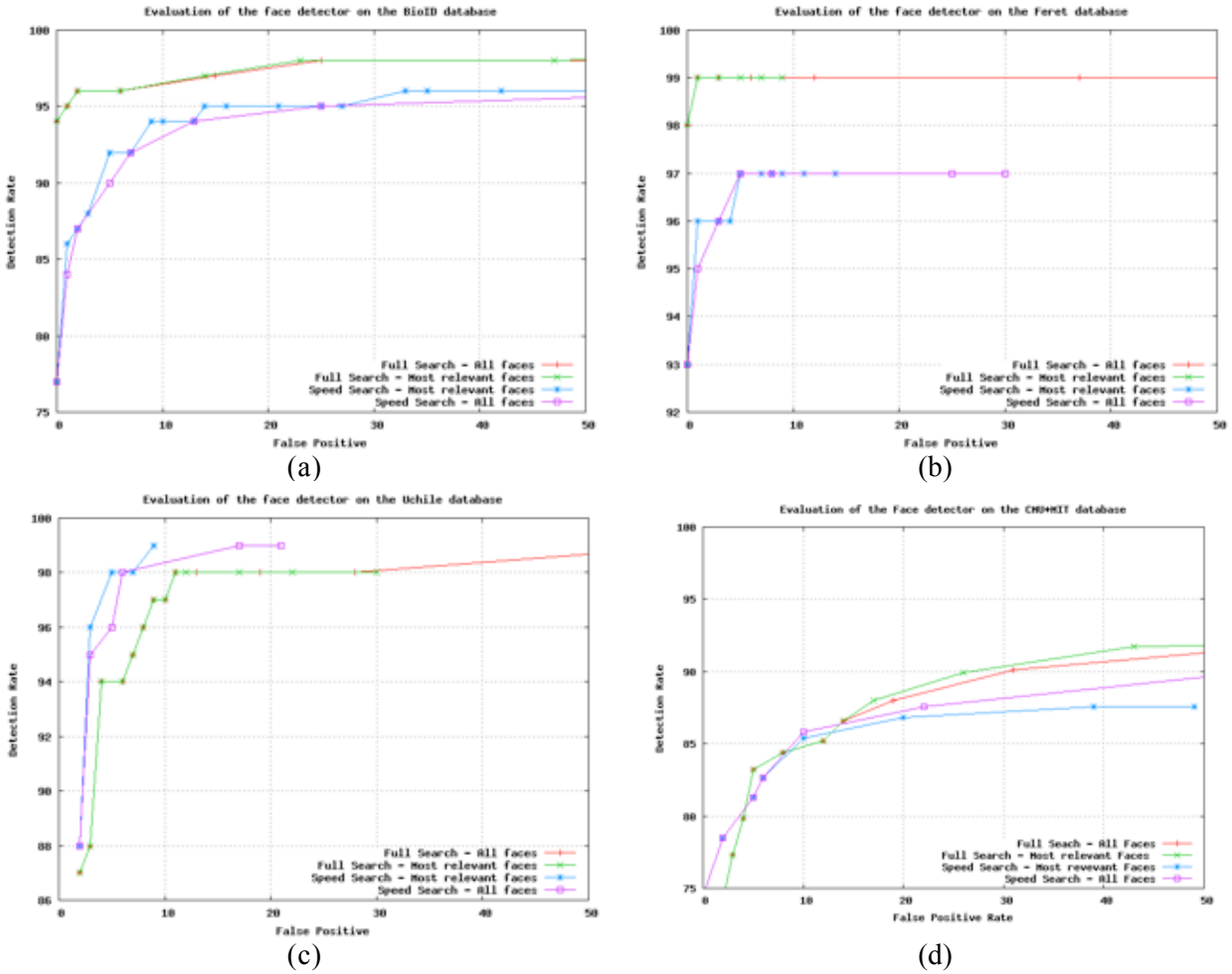


Figure 12. ROC curves of the different face detector flavors on the BioID (a), FERET (b), UCHFACE (c), and CMU-MIT databases.

Table 1: Comparative evaluation of the face detector on the BioID Database (1,521 images).

False Positives	**0**	**1**	**2**	**5**	**6**	**13**	**14**	**15**	**20**	**25**
Full-All	94.1	95.1	96.5		96.9			97.6		98.1
Full-Most	94.1	95.1	96.5		96.9		97.6			98.1
Speed-All	77.1	84.0	87.4	90.2		94.6				95.6
Speed-Most	77.1	86.1	88	92.6		94.6	95.1	95.2	95.3	95.6
Fröba [5]		~50		~64.5				~84	~94	~98

- Face detection in Single Face Images: FERET database

In figure 12(b) are shown the ROC curves of the different face detector flavors on the FERET database. Some selected points of these ROCs are shown in table 2. No other groups have reported face detection results on this subset of the FERET. However, we can affirm that the detection rate in the dataset is very high. From the 1,016 faces to be detected, our best performing algorithms, Full-All and Full-Most, detect 98.7% of them with 0 false positives and 99.5% with 1 false positive. These numbers are very good if we think on the potential application of a face recognition system after the face detection stage (FERET is a face recognition test set), and that we are using a face detection system, not a face localization system.

Table 2: Comparative evaluation of the face detector on the Feret Database (1,016 images).

False Positives	0	1	3	4	7	8	9	11	12
Full-All	98.7	99.5	99.7						99.7
Full-Most	98.7	99.5	99.6		99.7		99.8		
Speed-All	94	95.7	96.4			97.6			
Speed-Most	94	96.2	96.4	96.7	97.3	97.6	97.7	97.7	

- Face detection in Multiple Face Images: UCHFACE database

In figure 12(c) are shown the ROC curves of the different face detector flavors on the UCHFACE database. Some selected points of these ROCs are shown in table 3. No other groups have reported face detection results on this new database. However, considering that the images were obtained under uncontrolled conditions, and that they contain several faces per image, we consider that the obtained results are rather good (e.g. 96.5% with 3 false positives, 98.5% with 5 false positives).

Table 3: Comparative evaluation of the face detector on the UCHFACE (142 images, 343 faces).

False Positives	2	3	5	6	7	8	9	17
Full-All	87.8	88.0		94.8		96.5	97.1	98.5
Full-Most	87.8	88.0		94.8	95.9	96.5	97.1	98.5
Speed-All	88.6	95.6	96.8	98.5				99.1
Speed-Most	88.6	96.5	98.5		98.8		99.1	

- Face detection in Multiple Face Images: CMU-MIT database

In figure 12(d) are shown the ROC curves of the different face detector flavors applied to the CMU-MIT database. Because of some of these images are noisy and low quality, before applying our face detector to the CMU-MIT database, we have preprocessed them with a low pass filter to. The results here presented were obtained after applying the low pass filter.

Some selected points of these ROCs are shown in table 4 for comparing these results with the best results reported in the literature in this database. As it can be seen in the figure 12(d) all flavors have similar performance for low FP (false positives) values. However, for operation points with larger FP, the full search, most relevant faces flavor of the face detector gives better results. If we compare to other methodologies presented on the literature in terms of DR and FP, we obtain better results than [36][24], slightly better results than [17], slightly worse results than [6] (but our system is much faster[2]), and worse results than [40] and [29]. It is difficult to compare our system with [5], because they use a reduced subset of the CMU-MIT databases. But considering that using the complete database is more difficult, mainly because drawings of faces are removed from the original dataset, our results are better than the ones of [5]. We think we have lower detection rates than [40] and [29] mainly because of the size of the training database. As we have mentioned, our training database consists of 5,000 face images, while for example in [40] 20,000 training faces are employed.

Table 4: Comparative evaluation of the face detector on the CMU-MIT database (130 images, 507 faces).

False Positives	0	3	5	6	10	13	14	19	22	25	29	31	57	65
Full-All		77.3	83.2				86.6	88		89.9		90.1		92.1
Full-Most		77.3	83.2				86.6						92	
Speed-All	74.6		81.3	82.6	85.8				87.6				90.1	
Speed-Most	74.6		81.3	82.6	85.4			87						
Fröba [3] [5]	~66		~87							~90				
Wu 2004 [40]		89			90.1	90.7							94.5	
Viola&Jones					76.1							88.4		92

[2] That system is about 8 times slower than Viola&Jones. Our system has about the same processing speed than Viola&Jones.

[3] Subset of 483 from 507 faces. This set is called CMU 125 testset.

2001 [36]														
Rowley [24]					83.2							86		
Schneiderman [29]				89.7				93.1			94.4			
Schneiderman & Kanade 2000 [30]														94.4
Li. et al. [17]					83.6							90.2		
CFF 2004 [6]	88.8				90.5							91.5		92.3

4.2.3 Eye detection evaluation

There are a lot of eyes detection algorithms proposed in the last years, and we should select some of them for comparison purposes. For selecting state of the art eyes detection algorithms we ask to fulfill the following requirements: (i) they should be real time, (ii) a quantitatively characterized of them, using standard databases, should be available, and (iii) the evaluation databases should include complex background and variables illumination conditions. Two recently proposed systems that fulfill these requirements are [20] and [3]. Both of them use boosted classifiers, Adaboost and Gentleboost, respectively, and a previous stage of face detection. Both of them evaluated their eyes detector on the BioID database. For evaluating the eyes detection accuracy we employ cumulative curves of eyes localization errors as in [20] (the relative error of the detection is defined as the Euclidian distance between the ground truth of the eyes and of the centers of the detected eyes, normalized by the Euclidian distance between the ground truth eyes centers).

In figure 13 are shown these cumulative curves for the different eyes detector flavors[4] on the BioID (a), FERET (b), and UCHFACE (c) databases. No eyes detection experiments were performed on the MIT-CMU database because in many cases the resolution of the contained faces is too low for performing eyes detection. For evaluating the eye detection system, only correctly detected faces where used. Some selected points of these error cumulative curves are shown in table 5, together with the mean error in pixels for the eyes detection. It can be seen that the obtained results are very good: for example 4.4 pixels error and 99.4% DR on the BioID database. The average distance between the eyes for the BioID database is 54 pixels, so for a 99.4% DR the normalized error (in terms of the eye distance) is only 8%. In the case of the FERET images, for a 99.9% DR the normalized error is 9%. In the case of the UCHFACE images, for a 84.9% DR the normalized error is 9%. The table 5 shows more values in terms of the normalized error. It can also be noticed that the full search flavor gives slightly better results for the eye detector.

By looking at the cumulative error curves on the BioID database we observe that we obtain similar results than the ones reporter in [20]. There are just small differences that we do not consider statistically significant. In [3] a completely different methodology is employed for evaluating the performance of the eyes detector. No curves are given but median accuracy measured in terms of *irisis*. It is very difficult to compare this kind of results with ours. But, by analyzing the employed training methodology, and the eyes detection examples showed in that paper, we believe that those results are comparable with ours.

No other groups are reported eyes detection results on our subset of the FERET database or in the new UCHFACE database. We hope in a near future other research groups can employ these databases and the available ground truth for evaluating their systems.

[4] Our eyes detector is applied after face detection. Given that we have 4 different face detector flavors, we obtained also 4 different eyes detector flavors, depending on the employed face detector.

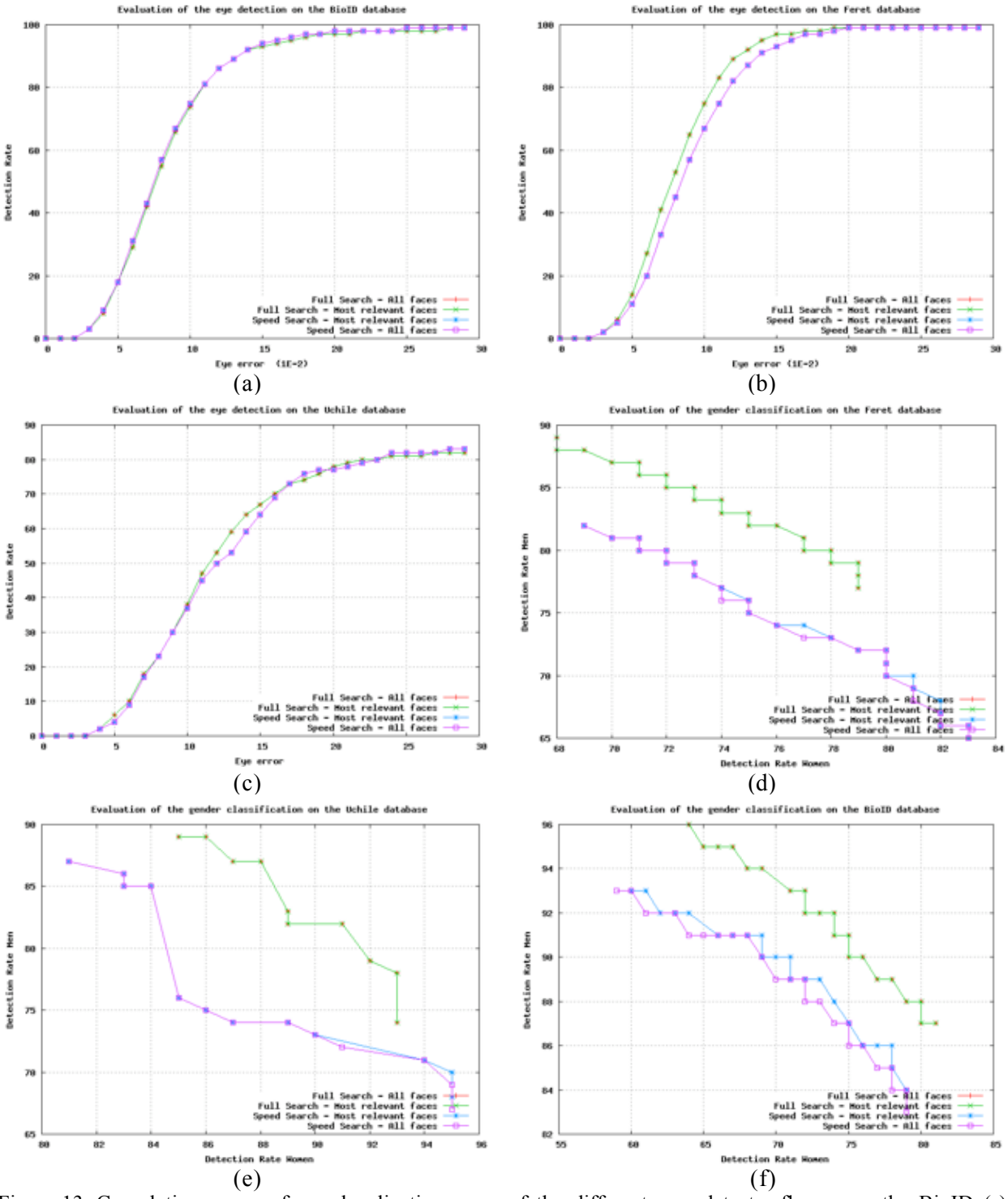


Figure 13. Cumulative curves of eyes localization errors of the different eyes detector flavors on the BioID (a), FERET (b), and UCHFACE (c) databases. Male detection rate versus female detection rate curves of the different gender classification flavors on the BioID (d), FERET (e), and UCHFACE (f) databases.

Table 5: Comparative evaluation of the eyes detection flavors on the BioID, FERET and UCHFACE databases.

BioID											
Speed	**Detection Rate**	67.1	75.0	81.4	92.8	94.5	95.7	97.8	98.5	99.0	99.4
	Normalized Error	0.06	0.06	0.07	0.07	0.08	0.08	0.08	0.08	0.08	0.08
	Mean Error in pixels	3.3	3.5	3.7	4.0	4.1	4.1	4.2	4.3	4.3	4.4
Full	**Detection Rate**	66.9	74.1	81.2	92.1	93.5	95.0	97.0	98.4	98.5	99.2
	Normalized Error	0.06	0.06	0.07	0.07	0.08	0.08	0.08	0.08	0.08	0.08
	Mean Error in pixels	3.4	3.5	3.7	4.0	4.1	4.2	4.3	4.4	4.4	4.4
Feret											
Speed	**Detection Rate**	57.2	67.0	76.0	91.1	93.8	95.5	98.6	99.6	99.9	99.9
	Normalized Error	0.06	0.07	0.07	0.08	0.08	0.08	0.09	0.09	0.09	0.09
	Mean Error in pixels	4.2	4.6	4.9	5.5	5.6	5.7	5.9	6.0	6.1	6.1
Full	**Detection Rate**	65.5	75.9	84.0	95.0	97.0	97.8	99.2	99.7	99.8	99.9
	Normalized Error	0.06	0.07	0.07	0.08	0.08	0.08	0.08	0.08	0.08	0.08
	Mean Error in pixels	4.2	4.5	4.8	5.2	5.3	5.3	5.5	5.5	5.5	5.5
UchileDB											
Speed	**Detection Rate**	47.4	54.7	60.1	74.3	77.0	79.2	81.9	84.0	84.3	84.9
	Normalized Error	0.06	0.07	0.07	0.08	0.08	0.08	0.09	0.09	0.09	0.09
	Mean Error in pixels	2.3	2.4	2.6	3.0	3.1	3.1	3.3	3.4	3.4	3.5
Full	**Detection Rate**	50.3	56.2	62.4	74.3	76.0	78.1	81.7	83.4	83.7	84.9
	Normalized Error	0.06	0.07	0.07	0.08	0.08	0.08	0.09	0.09	0.09	0.09
	Mean Error in pixels	2.4	2.5	2.6	2.9	3.0	3.0	3.2	3.3	3.3	3.4

4.2.4 Gender classification evaluation

For evaluating the gender classification accuracy we employ male detection rate (MDR) versus female detection rate (FDR) curves. In figure 13 are shown these curves for the different gender classification flavors[5] on the BioID (d), FERET (e), and UCHFACE (f) databases. No gender classification experiments were performed on the MIT-CMU database. Some selected points of the curves are shown in table 6 for future comparison with these results with the best results reported in the literature in these databases. It can be noticed that the full search flavor gives better results for the gender classifier. When using the full search the gender classifier has from 5% to 10% better classification rates compared to the speed search for the 3 databases. We think this is because the full search gives more accurate detections.

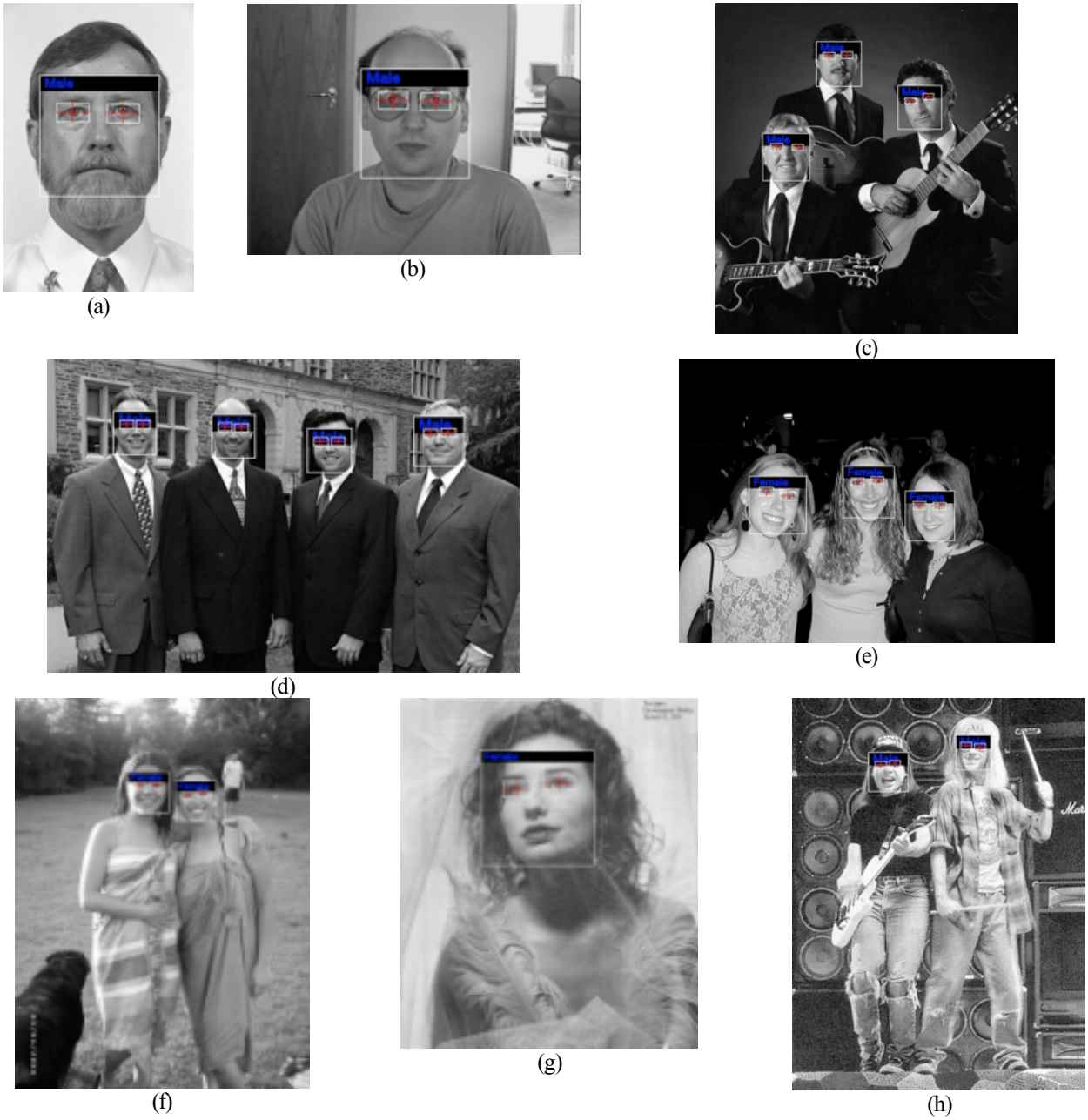
It is difficult to compare our gender classification system with other systems, mainly because many of the already proposed gender classification systems are applied to images with restricted background or illumination conditions, or because many of them are not real time. Just for having a reference, in [41] and [31] are proposed gender classification systems based on boosted classifiers. Both systems were evaluated in images downloaded from Internet and the obtained classification rates were 88% and 79%. Of course these results are not comparable between them, or with ours. However, we can see that the obtained results are of the same order than ours. All three systems are real-time.

Finally, some selected examples of our face detection, eyes detection and gender classification systems at work on the FERET, BioID, UCHFACE and MIT-CMU databases are shown in figure 14.

Table6: Comparative evaluation of the gender classification flavors on the BioID, FERET and UCHFACE databases. FDR: Female Detection Rate. MDR: Male Detection Rate.

BioID	**FDR**	64.5	67.1	69.9	72.6	75.2	78.2	81.0	82.0
	MDR	96.1	95.3	94.1	92.9	90.9	89.5	87.7	86.8
FERET	**FDR**	68.2	70.4	72.9	75.9	77.1	79.2	79.3	79.6
	MDR	89.2	87.9	85.5	82.7	81.8	79.3	78.4	77.3
UCHFACE	**FDR**	85.2	87.9	88.6	89.3	91.9	92.6	93.3	94.0
	MDR	89.7	87.4	85.1	83.3	82.2	79.9	78.7	73.6

[5] Our gender classification algorithm is applied after face detection. Given that we have 4 different face detector flavors, we obtained also 4 different gender classification algorithms.



(a) (b) (c) (d) (e) (f) (g) (h)

Figure 14. Some selected examples of our face detection, eyes detection and gender classification systems at work on the FERET (a), BioID (b), UCHFACE (c-f) and MIT-CMU (g-h) databases.

5. Human-Computer Interaction Applications

5.1. Multiple Face Detection and Tracking in Dynamic Environments

5.1.1. Face Tracking using Kalman Filtering

An essential skill of face-based human-computer interaction systems is the multiple detection and tracking of faces in dynamic environments. This skill is useful in applications like video conferencing, computer games and interfaces, mobile devices communication, video summarizing, customer profiling by observing attention, human-robot interaction, surveillance, and drivers monitoring, jut no name a few. In this section we will describe such an application, which is based on

the face detection system described in section 3.3. The tracking of the faces is based mainly on the use of Extended Kalman Filters (EKFs). Although from the theoretical point of view it can be argued that Particle Filters (e.g. [10]) are superior than EKF because of the Gaussianity hypothesis [7], our experience with self-localization algorithms for mobile robotics [16] tell us that the performance of both kind of filters in tracking and self-localization tasks is rather similar. Moreover, it is possible to obtain a very fast implementation of the EKF if the state vector is small, as in our case, because for each tracked object a different EKF is employed. This in very important when, as in our case, several objects are tracked at the same.

5.1.2. State Vectors and Parameters Database

Each object (a face) is characterized by its position in pixels in the frame, its width, its height, and the corresponding changing rates of these variables. The eight variables are the state vector of a first order EKF ($\mathbf{x}_k$). The parameters database (DB) stores the latest state vector ($\mathbf{x}_{k-1}$) for each object under tracking and its associated EKF. Since the detected features do not include the change rate components, these components are estimated as:

$$\mathbf{z}_k^{\mathbf{T}} = \left(z_k^1 \quad z_k^2 \quad z_k^3 \quad z_k^4 \quad \frac{z_k^1 - x_{k-1}^1}{\Delta t} \quad \frac{z_k^2 - x_{k-1}^2}{\Delta t} \quad \frac{z_k^3 - x_{k-1}^3}{\Delta t} \quad \frac{z_k^4 - x_{k-1}^4}{\Delta t} \right)^{\mathbf{T}}.$$

With z_k the vector of observations. The update model is:

$$\hat{\mathbf{x}}_k = \begin{pmatrix} 1 & 0 & 0 & 0 & \Delta t & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & \Delta t & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & \Delta t & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & \Delta t \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} \hat{\mathbf{x}}_{k-1}$$

5.1.3. Tracking Procedure

The block diagram of the multiple face detection and tracking system is shown in figure 15. Input images are analyzed in the *Face Detector* module, and detected faces are further processed by the *Detected-Tracked Object Matching* module. In this module the detected faces are matched with the current objects under tracking. Each new detection (a face window) is evaluated in the Gaussian function described by the state vector and its covariance matrix on the Kalman filter of each object. In this way, a matching probability is calculated. If the matching probability is over a certain threshold, then the detected face is associated with the corresponding object. If no object produces a probability value over that threshold, then the detected face is a new candidate object, and a new state vector (and Kalman filter) is created for this new object (*New Object Generator* module).

For each object under tacking, the prediction model estimates its a priori state (*Object State Prediction* module). Then, the a priori state is updated using all the detections associated with this state in the matching stage (*Object Update* module). If any candidate object accomplish the promote rule (over a certain amount of detections in a maximal amount of frames) then it become a true object (*Candidate Promoter* module). Finally, if a candidate object has more than a certain amount of frames with not enough associated detections (below a certain threshold), it is eliminated from the database (*Object Filter* module). True objects with state probability below a certain threshold are also eliminated from the database.

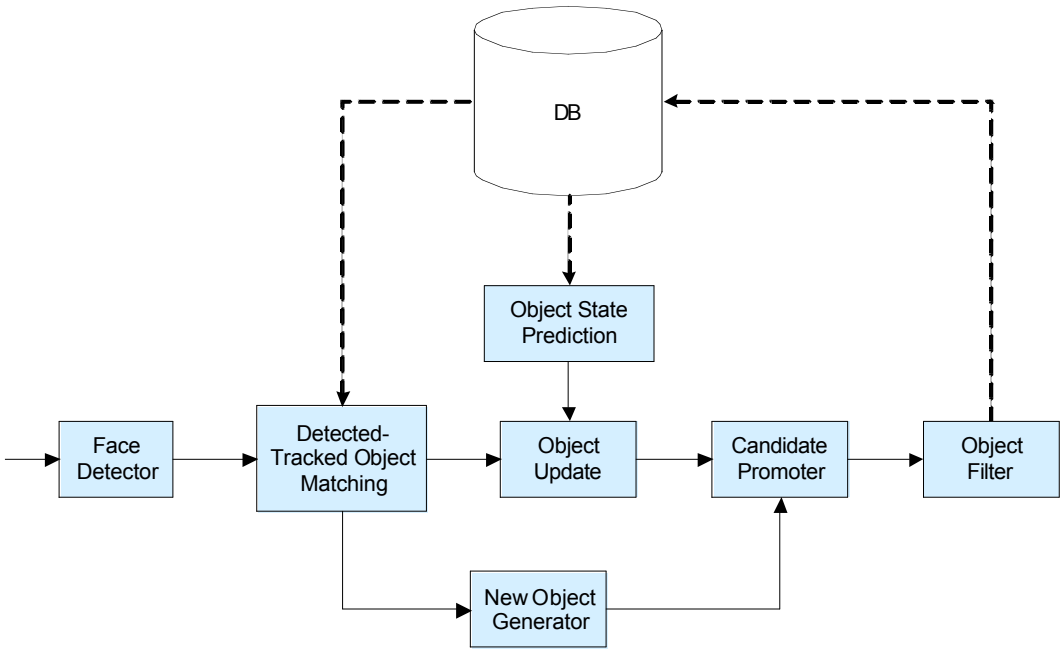


Figure 15. Block diagram of the multiple face detection and tracking system.

5.1.4. Face Detection and Tracking Evaluation

In order to test performance of our multiple face detection and tracking system we employed the PETS-ICVS 2003 dataset. We compared our detection and tracking results in terms of DR and FP with the ones obtained by other groups.

The PETS initiative corresponds to a very successful series of workshops on Performance Evaluation of Tracking and Surveillance. The PETS 2003 topic was gesture and action recognition, more specifically the annotation of a "smart meeting" (includes facial expressions, gaze and gesture/action). The PETS-ICVS 2003 dataset [46] consists of video sequences (frame from 640x480 pixels) captured by three cameras on a conference room. Two cameras (camera 1 and 2) were placed on opposite walls capturing the participant on each side of the room, and the third camera (camera 3) is an omnidirectional camera on the desk center. The dataset is divided in four scenarios A, B, C and D. For this analysis frames from scenarios A, B, and D, and cameras 1 and 2, were used. The ground truth consists of the eyes coordinates for those frames divisible by 10. In our experiments all frames were processed, but for statistics just two sets of images were considered: (i) Set A: all annotated frames, i.e. frames with frame number divisible by 10, and (ii) Set B: frames with frame number divisible by 100. The set A contains 49,350 frames and 10,308 annotated faces, while the set B contains 4,950 frames and 1,000 annotated faces. We included the test B because other research groups have tested their algorithms using this set.

In table 7 are shown the detection results obtained by our face detector on both sets. These results are much better than the ones reported in [2], where the Fröba-Kullbeck detector [4] and the Viola&Jones detector [36] were tested on the set B. In that test the Viola&Jones detector outperforms the Fröba-Kullbeck, but the results it obtains are very poor, 50% DR with 202 false positives or 62.2% DR with 2,287 false positives. We can conclude that our face detector performs very well in this real-world dataset (4,950 frames), and that the amount of FP is extremely low.

Table7: Face detection results on the sets A and B from PETS-ICVS 2003.

Set B	**Detection Rate**	67,2%	60,9%	53,5%	44,7%	37,9%
	False Positives	88	50	37	32	22
Set A	**Detection Rate**	67,9%	62,1%	50,8%	44,9%	36,2%
	False Positives	851	465	334	292	242

Then, we analyzed the performance of our tracking system and we quantify the improvement in the face detection process when using such a system. We analyzed the behavior of three different parameters for the tracking system:

- CTO (Candidate to true object) threshold: A new detection is immediately added to the database in order to track it, but it is not considered a true tracked object until CTO other detections are associated with it.
- MCF (Max candidate frames): If a candidate to object does not reach the CTO threshold in MCF frames since it was added to the database, then it is eliminated.
- Q: This is the covariance of the process noise in the Kalman filter.

In table 8 and 9 are shown the detection results after applying the tracking system on sets A and B. It can be seen that thanks to the tracking the number of FP decreases largely, up to 21% in set B and 35% in set A, and that at the same time the DR increases.

Table8: Face detection results, after tracking, on set B from PETS-ICVS 2003.

Tracking Parameters			**False Positive**	**Detection Rate**	**False Positive Decrement**	**Detection Rate Increment**
CTO	**MCF**	**Q**				
3	5	1.0	69	0,689	21,6%	2,5%
3	7	1.0	71	0,689	19,3%	2,5%
2	2	1.0	71	0,689	19,3%	2,5%
1	2	2.0	72	0,68	18,2%	1,2%
1	5	2.0	76	0,681	13,6%	1,3%
1	7	2.0	76	0,681	13,6%	1,3%
2	5	1.0	80	0,698	9,1%	3,9%
2	7	1.0	80	0,699	9,1%	4,0%
3	5	0.5	85	0,707	3,4%	5,2%
2	2	0.5	87	0,706	1,1%	5,1%
3	7	0.5	88	0,707	0,0%	5,2%

Table9: Face detection results, after tracking, on set A from PETS-ICVS 2003.

Tracking Parameters			**False Positive**	**Detection Rate**	**False Positive Decrement**	**Detection Rate Increment**
MCF	**MCF**					
3	5	2.0	525	0,651012799	35,0%	10,7%
3	7	2.0	530	0,652322886	34,4%	11,0%
2	2	2.0	536	0,657059357	33,7%	11,8%
2	5	2.0	580	0,667640834	28,2%	13,6%
2	7	2.0	580	0,667640834	28,2%	13,6%
3	5	1.0	625	0,685075078	22,6%	16,5%
3	7	1.0	655	0,689609997	18,9%	17,3%
2	2	1.0	629	0,687392926	22,2%	16,9%
1	2	2.0	683	0,683059559	15,5%	16,2%
1	5	2.0	700	0,685881286	13,4%	16,7%
1	7	2.0	700	0,685881286	13,4%	16,7%
2	5	1.0	738	0,700997682	8,7%	19,2%
2	7	1.0	750	0,70230777	7,2%	19,5%

For demonstrative purposed we have processed a set of video-conference like videos, that can be downloaded from [47]. These videos were processed by our multiple face detection and tracking system. Eyes detection and gender classification was also included in the analysis.

5.2. Person Detection and Tracking for AIBO Robots

Sony AIBO robots correspond to one of most widespread and popular personal robots. As all of us know, thousand from children and researchers employ AIBO robots for entertainment or research. We do believe that in a near future personal robots will be far more widespread than today. One of the basic skills that personal robots should integrate is the face-based visual interaction with humans. Robust face detection and tracking is key step in this direction. For implementing such a system we adapted the multiple face detection and tracking system described in section 5.1 for Sony AIBO robots model ERS7.

ERS7 robots have a 64bit RISC Processor (MIPS R7000) from 576 MHz, 64MB RAM and a color camera of 416x320 pixels that delivers 30fps. The face and tracking detection system was integrated with our robot control library U-Chile1 [27][16]. U-Chile1 is divided in five task-oriented modules: *Vision*, which contains mostly low-level vision algorithms, *Localization*, in charge of the robot self-localization, *Low-level Strategy*, in charge of the behavior-based control of the robots, *High-level Strategy*, in charge of the high level robot behavior and strategy, and *Motion Control*, in charge of the control of the robot movements. U-Chile1 runs in real-time and after the integration with the tracking system we were able of running our face detection and tracking system at a rate of 2fps. We included also eyes detection and gender classification. The eyes detection system works when the resolution of the faces is larger than 50x50 pixels.

In figure 16 are shown some selected examples of face detection and tracking using an AIBO ERS7. The system detects faces, gender classification and eyes detection. More examples can be seen in [47].

Figure 16. Examples of the face detection and tracking system for AIBO robots. The system detect faces and performs gender classification. When the resolution of the faces is larger than 50x50 pixels it detects the also the eyes.

5.3. Tools for Supporting Content-based Image Characterization on Specific Web Collections

Human-computer interfaces should make easier and natural the interaction between humans and computers, but it should also allow the automatic characterization of large multimedia databases (like Internet and personal image collections), the search and retrieval on them, and even the automatic filtering of objectionable contents in those databases. If we take the Web as a case study, we will see the following. Given the trend to enrich websites with multimedia, it becomes increasingly important to be able to characterize a given segment of the Web according to the multimedia elements that it

contains. This type of information is of great importance for Internet service providers (who can determine required levels of regional service), for content producers, for researchers in content-based retrieval, and for web search application developers. However, characterizing the multimedia contents of the Web, is a challenging technical problem. First, one must deal with huge amounts of distributed data. Second, it is necessary to use media-specific content-based analysis tools to be able to determine the content of multimedia data (i.e., not just using metadata). With images and video, this means developing tools to automatically determine their visual characteristics using features that represent color, texture, shape, etc. More interestingly, it implies using algorithms to automatically detect objects of interest (e.g. persons). Obviously, given the large amounts of data, manual characterization of Web content is not an option.

Another very important problem in the web, and in general in the Internet, is the presence of harmful (e.g. pornographic) or even illegal (e.g. pedophilic) contents. The amount of this non-desired material is growing at an increasing rate. Therefore, the development of automatic filters for non-desired material based on image contents is a must. However, filtering is not the only possible action to be taken in order to deal with the increasing amount of pornographic material in the Web. On the one hand, the automatic pornographic sites' detection using content-based analysis can allow the discovering of new pornographic sites that can be added to lists of prohibited sites, which can be then filtered using traditional methods or even detailed analyzed for law enforcement purposes. On the other hand, a characterization of the pornographic contents in a given segment of the web can be important for determining users' behaviors and user bandwidth requirements in the case that only nude images are processed.

In this context, we have developed tools for supporting the content-based image characterization of specific Web collections using face information, among others. Face information is very important in these characterization studies. In some cases this is the only reliable information for determining the presence of persons in the images under analysis. First, in [11] we developed a system for characterizing the image content of the .CL web segment. This characterization was made using some basic low-level features (color, edges and textures), skin detection and face detection. Face detection was implemented using the same face detector already described in this work. Some of the obtained results are the following: about 2.07% of the HP- (Home Page) images and 2.12% of the IP- (Inner Pages) images contained faces; the average number of faces per image (from those images containing faces) is 2.1167/2.1162 for the HP- and IP-images, respectively. It was also found that the distribution of the number of faces in both image sets (considering only the images that contain faces) is close to a Power law.

Second, in [25] we performed another study in which we analyzed the pornographic content of the same web segment, using the same processing tools. Some of the most interesting obtained results, related with face contents, are the following: about 40% of porno and 36% of nude images contains at least one frontal face. Moreover, an important number of adult images contain just one or two faces (31% nude and 33% porno).

The same methodology employed for characterizing the .CL web segment (see details in [11] and [25]) can be applied for carrying out similar studies in other segments of the web or in large multimedia databases, and it can also be used for the development of image retrieval systems.

6. Conclusions

Face analysis plays an important role for building human-computer interfaces that allow humans to interact with computational systems in a natural way. Face information is by far, the most used visual cue employed by humans. In the present work we have described a unified framework for the analysis of faces, with a special emphasis on the description of the training procedure. This framework is used for the development of a faces detector, an eye detector and a gender classifier. The system can work in real time, it has a high accuracy, and it has been successfully used in different applications. The presented framework could be also used for the detection of other objects, and for the classification of faces in terms of race, presence of glasses, etc. The most interesting aspect of this framework is the possibility of building classification/detection systems with high accuracy, robustness, high processing speed, and training speed.

This framework is based on the use of Adaboost for the training of a nested cascade classifiers, and the use of domain-partitioning for the design of the weak classifiers, which are based on rectangular and LBP features. For the training of the detection systems, internal and external bootstrapping is used, which together with other heuristics allows us to train a face detection system in about 15 hours using in a Pentium 4 personal computer. This training time is at least one order of magnitude lower than previously published training time for similar systems.

We have compared our face detection, eyes detection and gender classification systems with state of the art similar systems. Our face detection system obtains the best reported results on the BioID database, the best reported results on the PETS-ICVS 2003 dataset, and the third best reported results on the CMU-MIT database. Our eyes detection system obtains similar results than the best reported results on the BioID database. The gender classification obtains comparable results to the best ones reported in the literature, although not using the same database. For future comparisons with the here presented systems, a performance evaluation was performed on a subset of the FERET database and on the new UCHFACE database. This new database includes the annotation of the faces, eyes (plus other landmarks) and gender, and it has been made available for the research community.

The developed face detection, eyes detection and gender classification systems have been successfully used in the construction of human computer interfaces. First, we have integrated the face detector with a new tracking system, building a system for the tracking of multiple faces in dynamic environments. This system is able to detect and track faces with a high performance in real-world videos, and with an extremely low number of false positives compared to state of the art methodologies. Second, we integrated out face analysis and tracking system into the Sony AIBO robots. In this way the AIBO robots can interact with persons using the human faces and the gender and eye information extracted from it. Third, using our detection and classification systems we developed tools for the content-based image characterization of multimedia databases, which we used for characterizing specific web collections and for the filtering of objectionable images. Beside the already mentioned projects, we are currently applying the face analysis system described here in the development of an image retrieval tool for searching persons on Internet, and on a hand gesture recognition system.

In a near future we plan to use the proposed learning framework for building other tools for the analysis of faces (race classification, mouth detection, etc.), and for the detection of other objects (for examples cars). We plant to extend the framework to detect faces with in-plane and out-of-plane rotations, and to accurately estimate their poses. We also want to explore the use of other kinds of features and the use of LUT classifiers with partitions of variable size.

Acknowledgements

This research was funded by Millenium Nucleus Center for Web Research, Grant P04-067-F, Chile. Portions of the research in this work use the FERET database of facial images collected under the FERET program.

References

[1] S. Baluja, M. Sahami, H.A. Rowley, “Efficient face orientation discrimination”, *Int. Conf. on Image Processing, - ICIP '04*, Vol. 1, pp. 589 – 592, 24-27 Oct. 2004.

[2] D. Cristinacce and T. Cootes, “A Comparison of two Real-Time Face Detection Methods”, *4th IEEE Int. Workshop on Performance Evaluation of Tracking and Surveillance – PETS 2003*, pp. 1-8, Graz, Austria, April, 2003.

[3] I. Fasel, B. Fortenberry, and J. Movellan, “A generative framework for real time object detection and classification”, *Computer Vision and Image Understanding* 98 (2005) 182–210.

[4] B. Fröba and Ch. Küblbeck. Robust face detection at video frame rate based on edge orientation features. *5th Int. Conf. on Automatic Face and Gesture Recognition FG - 2002*, pp. 342–347, 2002.

[5] B. Fröba and A. Ernst, “Face detection with the modified census transform”, *6th Int. Conf. on Face and Gesture Recognition - FG 2004*, pp. 91–96, Seoul, Korea, May 2004.

[6] M. Delakis and C. Garcia, "Convolutional face finder: A neural architecture for fast and robust face detection", *IEEE Trans. Pattern Anal. Mach. Intell.*, 26(11):1408 – 1423, 2004.
[7] G. Dudek, and M. Jenkin, *Computational Principles of Mobile Robotics*, Cambridge University Press, 2000.
[8] E. Hjelmås, B. K. Low, "Face detection: A survey", *Computer Vision and Image Understanding* 83, 236-274, 2001.
[9] L.-L. Huang, A. Shimizu, and H. Kobatake, "Classification-based face detection using Gabor filter features", *6th Int. Conf. on Face and Gesture Recognition - FG 2004*, pp. 397 - 402, Seoul, Korea, May 2004.
[10] M. Isard, and A. Blake, "Condensation – Conditional Density Propagation for Visual Tracking", *Int. J. of Computer Vision*, Vol. 29, N. 1, pp. 5—28, 1998.
[11] A. Jaimes, J. Ruiz-del-Solar, R. Verschae, R. Baeza-Yates, C. Castillo, D. Yaksic, and E. Davis, "On the image content of a Web Segment: Chile as a case study", *Journal of Web Engineering*, Vol. 3, No.2 (2004) 153-168, Rinton Press.
[12] O. Jesorsky, K.J. Kirchberg, and R.W. Frischholz, "Robust Face Detection using the Hausdorff Distance", *3rd Int. Conf. on Audio- and Video-based Biometric Person Authentication – AVBPA 2001*, (*Lecture Notes in Computer Science, LNCS-2091*), pp. 90–95, Halmstad, Sweden, 6–8 June 2001.
[13] M. J. Jones and P. Viola, "Face Recognition Using Boosted Local Features", Mitsubishi Electric Research Laboratories, technical report TR2003-25, April 2003, available August 2005 in http://www.merl.com/publications/TR2003-025/.
[14] K.J. Kirchberg, O. Jesorsky, and R.W.Frischholz, "Genetic Model Optimization for Hausdorff Distance-Based Face Localization", *Int. ECCV 2002 Workshop on Biometric Authentication* (Lecture Notes In Computer Science; Vol. 2359), pp. 103 – 111, Copenhagen, Denmark.
[15] M. Kölsch and M. Turk, "Robust Hand Detection", *6th Int. Conf. on Face and Gesture Recognition - FG 2004*, pp. 614 - 619, Seoul, Korea, May 2004.
[16] R. Lastra, P. Vallejos, and J. Ruiz-del-Solar, "Integrated Self-Localization and Ball Tracking in the Four-Legged Robot Soccer League", *1st IEEE Latin American Robotics Symposium – LARS 2004*, pp. 54 – 59, Oct. 28 – 29, México City, México, 2004.
[17] S.Z. Li, L. Zhu, Z.Q. Zhang, A. Blake, H.J. Zhang, H. Shum, "Statistical Learning of Multi-view Face Detection", *7th European Conf. on Computer Vision – ECCV 2002*, (Lecture Notes in Computer Science; Vol. 2353), pp. 67 – 81, 2002.
[18] Y.-Y. Lin, T.-L. Liu, and C.-S. Fuh, "Fast object detection with occlusion", *8th European Conf. on Computer Vision - ECCV 2004* (*Lecture Notes in Computer Science 3021*), pp. 402-413, Prague, Czech Republic, May 11-14, 2004.
[19] C. Liu and H.-Y. Shum, "Kullback-leibler boosting", *IEEE Conference of Computer Vision and Pattern Recognition – CVPR 2003*, pp. 587-594, 2003.
[20] Y. Ma, X. Ding, Z. Wang and N. Wang, "Robust precise eye location under probabilistic framework", *6th Int. Conf. on Face and Gesture Recognition - FG 2004*, pp. 339 - 344, Seoul, Korea, May 2004.
[21] E. Osuna, R. Freund, and F. Girosi, "Training support vector machines: an application to face detection" *IEEE Conference of Computer Vision and Pattern Recognition – CVPR 97*, pp. 130 - 136, 1997.
[22] P. J. Phillips, H. Wechsler, J. Huang and P. Rauss, "The FERET database and evaluation procedure for face recognition algorithms," *Image and Vision Computing J.*, Vol. 16, no. 5, pp. 295-306, 1998.
[23] M-H. Yang, D. Roth, and N. Ahuja, "A SNoW-Based Face Detector", *Advances in Neural Information Processing Systems 12*, S.A. Solla, T.K. Leen, and K.-R. Müller, (eds.), pp. 855-861, 2000.
[24] H. Rowley, S. Baluja, and T. Kanade, " Neural Network-Based Detection", *IEEE Trans. Pattern Anal. Mach. Intell.,* Vol.20, No. 1, 23-28, 1998.
[25] J. Ruiz-del-Solar, V. Castañeda, R. Verschae, R. Baeza-Yates, and F. Ortiz, "Characterizing Objectionable Image Content (Pornography and Nude Images) of specific Web Segments: Chile as

a case study", *3rd IEEE Latin American Web Congress - LA-WEB 2005*, Buenos Aires, Argentina (in press).
[26] J. Ruiz-del-Solar, and Navarrete, "Eigenspace-based Face Recognition: A comparative study of different approaches", *IEEE Trans. on Systems, Man, and Cybernetics C* (Special Issue on Biometric Systems), Vol. 35, No. 3, pp. 315 – 325, Aug. 2005.
[27] J. Ruiz-del-Solar, P. Vallejos, R. Lastra, P. Loncomilla, J.C. Zagal, C. Morán, and I. Sarmiento, "UChile1 2005 Team Description Paper", RoboCup 2005 Symposium, July 13 – 17, Osaka, Japan (CD Proceedings).
[28] R.E. Schapire and Y. Singer, Improved Boosting Algorithms using Confidence-rated Predictions, *Machine Learning*, 37(3):297-336, 1999.
[29] Henry Schneiderman, Feature-Centric Evaluation for Efficient Cascade Object Detection, *IEEE Conference of Computer Vision and Pattern Recognition – CVPR 2004*, pp. 29 - 36, 2004.
[30] H. Schneidermann and T. Kanade, " A statistical model for 3D object detection applied to faces and cars", *IEEE Conf. on Computer Vision and Pattern Recognition*, Vol. 1, pp. 746 – 751, 2000.
[31] G. Shakhnarovich, P. Viola & B. Moghaddam, A Unified Learning Framework for Real Time Face Detection & Classification, *Int Conf. on Automatic Face & Gesture Recognition - FG'02*, pp. 16 – 26, May 2002.
[32] K. Sung and T. Poggio, "Example-Based Learning for Viewed-Based Human Face Deteccion", *IEEE Trans. Pattern Anal. Mach. Intell.*, Vol.20, No. 1, 39-51, 1998.
[33] A. Torrealba, K.P. Murphy, and W.T. Freeman, "Sharing Visual Features for Multiclass and Multiview Object Detection", AI Memo 2004-008, April 2004, MIT.
[34] R. Verschae, and J. Ruiz-del-Solar, "A Hybrid Face Detector based on an Asymmetrical Adaboost Cascade Detector and a Wavelet-Bayesian-Detector", Lecture Notes in Computer Science 2686 (IWANN 2003), Springer, pp. 742-749, Menorca, Spain.
[35] R. Verschae, J. Ruiz-del-Solar, M. Köppen, R.V. Gracia, "Improvement of a Face Detection System by Evolutionary Multi-Objective Optimization", *5th Int. Conf. on Hybrid Intelligent Systems - HIS 2005*, Nov. 6-9, 2005, Rio de Janeiro, Brazil (in press).
[36] P. Viola and M. Jones, "Rapid object detection using a boosted cascade of simple features", *IEEE Conf. on Computer Vision and Pattern Recognition - CVPR*, Kauai, HI, USA, 2001, pp. 511 - 518.
[37] P. Viola and M. Jones, "Fast and robust classification using asymmetric adaboost and a detector cascade", *Advances in Neural Information Processing System 14*. MIT Press, 2002.
[38] Viola & Jones CVPR 2003 (Multiview)
[39] P. Viola, M. Jones, and D. Snow, "Detecting Pedestrians Using Patterns of Motion and Appearance", *9th IEEE Int. Conf. on Computer Vision – ICCV 2003*, Vol. 2, pp. 734 - .741, Oct. 13 - 16, 2003.
[40] B. Wu, H. AI, C. Huang, and S. Lao, "Fast rotation invariant multi-view face detection based on real Adaboost", *6th Int. Conf. on Face and Gesture Recognition - FG 2004*, pp. 79–84, Seoul, Korea, May 2004.
[41] Bo WU, Haizhou AI, Chang HUANG. "LUT-based Adaboost for Gender Classification" *In: 4th International Conference on Audio and Video-based Biometric Person Authentication*, June 10-11, 2003, Guildford, United Kingdom
[42] R. Xiao, L. Zhu, H.-J. Zhang, "Boosting chain learning for object detection", *9th IEEE Int. Conf. on Computer Vision – ICCV 2003*, Vol. 1, pp. 709- 715, 13-16 Oct. 2003.
[43] M. Yang, N. Ahuja, and D. Kriegman, "Mixtures of linear Subspaces for Face Detection", *Fourth IEEE Int. Conf. on Automatic Face and Gesture Recognition*, pp. 70 – 76, 2000.
[44] M. Yang, D. Kriegman, N. Ahuja, "Detecting Faces in Images: A Survey", *IEEE Trans. Pattern Analysis and Machine Intelligence*, Vol. 24, No 1, pp. 34-58, 2002.
[45] BioID Face Database. Available on August 2005 in: http://www.humanscan.de/support/downloads/facedb.php
[46] PETS2003 Home Page. Available on August 2005 in: http://www.cvg.cs.rdg.ac.uk/PETS-ICVS/pets-icvs-db.html

[47] Computational Vision Group, Universidad de Chile. Available on August 2005 in: http://vision.die.uchile.cl/