Machine Vision and Applications (2008) 19:85–103
DOI 10.1007/s00138-007-0084-0

ORIGINAL PAPER

A unified learning framework for object detection and classification using nested cascades of boosted classifiers

Rodrigo Verschae · Javier Ruiz-del-Solar · Mauricio Correa

Received: 30 August 2006 / Accepted: 5 March 2007 / Published online: 9 October 2007

Abstract In this paper a unified learning framework for object detection and classification using nested cascades of boosted classifiers is proposed. The most interesting aspect of this framework is the integration of powerful learning capabilities together with effective training procedures, which allows building detection and classification systems with high accuracy, robustness, processing speed, and training speed. The proposed framework allows us to build state of the art face detection, eyes detection, and gender classification systems. The performance of these systems is validated and analyzed using standard face databases (BioID, FERET and CMU-MIT), and a new face database (UCHFACE).

Keywords Object detection · Boosting · Nested cascade classifiers · Face detection · Eyes detection

1 Introduction

An important goal of machine vision is to develop systems that can detect objects in cluttered backgrounds, with the ability of generalization across intra-class variability. Among many other detection problems, face detection has concentrated a lot of attention in the last years, mainly because it is a key step in almost any computational task related with the analysis of faces in digital images, and in many different situations it is the only way to detect persons in a given scene. Face detection is a very challenging task, which should be performed robustly and efficiently, regardless of variability in scale, location, orientation, pose, illumination and facial expressions, and considering possible object occlusions. In many applications the real-time requirement is added. Because of these extreme requirements, many object detection methodologies have been proposed in the context of face detection. In addition, it should be noted that object detection is a particular kind of 2-class classification, in which there is a high asymmetry in the probability of occurrence of the classes (object vs. non-object), and therefore object detection methodologies can be also used for solving classification problems with more than two classes.

In this general context the aim of this work is to introduce a unified learning framework for object detection and classification using nested cascades of boosted classifiers. The most interesting aspect of this framework is the integration of powerful learning capabilities together with effective training procedures, which allows building detection and classification systems with high: (i) accuracy (high detection rates with a few false positives), (ii) robustness (operation in dynamical environments), (iii) processing speed (real-time), and (iv) training speed (a few hours in a standard PC). We apply the proposed framework to face detection, eyes detection and face classification problems. This learning framework can be used also for the construction of detection systems for specific objects' view/poses (e.g., hands, heads, pedestrians) and for building (few-classes) classification systems (e.g., race, age).

For a complex learning machine, having powerful learning capabilities without having adequate training procedures and data is useless. Consequently, the training procedures are as important as the learning algorithm. The proposed framework integrates both aspects, and addresses also the computational complexity aspects of the training. Key concepts of this framework are boosting, nested cascade classification, and bootstrap training:

R. Verschae (✉) · J. Ruiz-del-Solar · M. Correa
Department of Electrical Engineering, Universidad de Chile,
Av. Tupper 2007, 837-0451 Santiago, Chile
e-mail: rverscha@ing.uchile.cl

J. Ruiz-del-Solar
e-mail: jruizd@ing.uchile.cl

- Boosting is employed for finding (i) highly accurate hypotheses (classification rules) by combining several weak hypotheses (classifiers), and (ii) self-rated confidence values that estimate the reliability of each prediction. In particular, we use the so-called *domain-partitioning weak hypotheses* learning paradigm [21].
- Cascade classifiers consist of several layers (stages) of classifiers of increasing complexity for obtaining fast processing speed [31] together with high accuracy. This is possible because of two reasons: (1) there is an important difference in the a priori probability of occurrence of the classes, i.e., there are much more non-object than object windows[1] in any given image, 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. Hence the average processing time of a window is almost completely defined by the expected processing time of non-object windows. Nested cascade classification [35] allows obtaining higher classification accuracy by reusing in each layer the confidence given by its predecessor.
- Special attention should be given to the selection of the training examples and to the training procedure. When working with discriminative methods, it is important to have training samples that correctly define the classification boundary, in particular to use non-object patterns that look similar to the objects. This selection can be done using the bootstrap procedure [26], which basically consists in to re-train iteratively a classifier, adding in each iteration the incorrectly classified examples to the training set. When training a cascade classifier an important question that arises is which (kind of) non-object examples should be used to train each stage of the cascade. For this, we extend the procedure used by [31], and we apply bootstrap not only before the training of each layer, but also during the training of each layer.

Besides taking and putting together the best ideas from state of the art works in cascade-based object detection, our most important contributions are mainly focused on the successful strategy for training cascades of boosted classifiers. An important point to consider is the computational complexity of the training process of the cascade. For example, in a system like the one presented in [32], the training time can take several months on a single Pentium 4 computer. One of the major advantages of our system is its reduced training time; it takes about 15 h in a standard PC, and it grows linearly with the number of training examples and with the number of features. 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 a bias value for forcing a low number of features in each stage, (5) the use of feature sampling, and (6) the use of features that can be evaluated very fast. Some of these aspects also allow obtaining high classification speed.

This article is structured as follows. In Sect. 2 some related works to the topics of boosting, cascade classification, and face detection are outlined. In Sect. 3 the proposed learning framework is presented, with special emphasis in the description of the training issues. In addition, face detection and eyes detection systems, built using this framework are described. In Sect. 4 is presented a comparative analysis of the critical components of the proposed learning framework, and a comparison of the trained face and eyes detection systems with state of the art systems using standard evaluation databases. In addition, we show the performance of a gender classification system built using the same framework. Finally, some conclusions and projections of this work are given in Sect. 5.

2 Related work

As mentioned, key concepts of the proposed framework are boosting, nested cascade classification, and bootstrap training. Many of these concepts have been developed within the context of face detection. Therefore, related works, in the face detection context, will be outlined in the following paragraphs.

Several approaches have been proposed for the computational detection of faces in digital images. A very comprehensive review can be found in [8] and [41]. 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 [8,11]. Starting with the seminal works of Rowley et al. [19] and Sung and Poggio [26] successful image-based approaches include the use of neural networks [4], SNoW classifiers [40], Bayesian classifiers [22,23,39], SVM [17], and boosted cascades [14,32,35]. This last approach is based on a learning paradigm introduced in Viola and Jones [31], which consists of using a cascade of boosted classifiers to obtain a very fast system that is capable at the same time of achieving high detection rates. They used Adaboost [21] as boosting algorithm, which has been widely used in different kinds of classifications problems because of its simplicity, high performance, and fast speed.

The boosted cascade paradigm has been widely studied and extended in the last few years, being [6,13,14,22,35,36]

[1] We call windows, the processing image regions analyzed by the classifier. They have a fixed size, normally between 19×19 and 25×25 pixels.

some of the most important works in this area, with some of them achieving the best reported results in face detection. The key idea for achieving fast detections is that the complexity of the classifiers (i.e., number of features in each classifier) increases when advancing in the cascade. Windows (image regions) that are easy to be classified as non-faces are discarded in the first stages (short processing time), while windows containing faces or face-like objects are analyzed by several cascade stages.

The cascade system proposed in [32] 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. The system developed by Wu et al. [35] uses domain-partitioning weak classifiers (originally proposed in [21]), which, compared to [32], achieves an important improvement in the representation power of the weak classifiers; it keeps a simple representation that at the same time increases the accuracy of the classification, and reduces the processing and training time (see details on Sect. 3.1). Another interesting idea proposed in [35], and also in [38], is the use of *nested* cascades. Nested cascades are cascades that use the confidence output of a given layer in the next layer of the cascade. This allows obtaining more compact (faster) cascades and more accurate classifications. Fröba and Ernst [6] introduce the mLBP (modified Local Binary Patterns) features, which are robust to extreme illumination conditions, and also the use of what we call internal bootstrap, for the cascade's training (see Sect. 3 for details). They use a cascade consisting of four layers, the first three layers correspond to boosted classifiers, and the last one is a SNoW classifier [39], all of them based on mLBP features.

An important drawback of the system proposed by [32] is the long training time. It can be up to several months if it is done in a single computer. Although computers are becoming faster (Moore's Law) the training time is still an important issue. Few works [2,28,37] have tried to overcome this problem when training cascade classifiers. Wu et al. [37] propose to replace the feature selection done by Adaboost by a Forward Feature Selection (FFS) procedure. The FFS can be performed before running Adaboost or it can replace completely Adaboost. Brubaker et al. [2] compare FFS to other three methods that select a subset of features prior to running Adaboost. One of the methods (RND) performs a random sampling of the features, the second method (RANK) ranks the features by mutual information, and the third method (CMI) ranks the features by conditional mutual information. They show that RND outperforms RANK, and that FFS and CMIM have similar performance, both outperforming RND. Verschae and Ruiz-del-Solar [28] and Baluja et al. [1] propose to sample the feature set before each iteration of Adaboost. In this way the training time is reduced proportionally to the percentage of features considered at each iteration, and Adaboost has the chance to select the features from a large set allowing a large diversity of the select features.

For obtaining an optimal cascade classifier in terms of processing time, false positive rate (FPR) and true positive rate (TPR), an important issue is how to handle the tradeoff between the number of features, the FPR and the TPR of the layer. In [25] a cost function to be minimized during the training of each layer is defined. This cost function is a sum of two terms; the first term corresponds to the cascade's TPR, which considers the desired target FPR of the complete cascade. The second term considers the expected number of features to be evaluated up to that layer multiplied by a parameter used to adjust the trade off between the two terms. The authors do not specify how to select this parameter. The procedure also needs to set the target FPR of the cascade. In [2], the selection of the number of features and the bias at each layer is also performed by minimizing a cost function, that takes into account the probabilities of achieving the desired TPR and FPR for the cascade.

It is worth mentioning that cascade classifiers have been also used in the detection of other kind of objects, like cars [22], hands [11], pedestrians [33], and traffic signs [22]. Regarding multi-class classification, in [27] the design of boosted classifiers for detecting different classes of objects in cluttered scenes is addressed by finding common features and weak classifiers that can be shared across the classes and/or views. Although they do not use cascade classifiers, we believe that the idea of sharing features could be incorporated in the here-proposed paradigm.

Most of the described concepts related with the design, construction and training of boosted cascades 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 in [32], with the later improvements introduced in [6] and [35]. Our most important contributions over previous work are mainly focused on the adequate training of the nested cascade of boosted classifiers, which also allows to considerably reducing the training time. These contributions include the use of internal and external bootstrap for training the layers of the cascade, the use of feature sampling before each iteration of Adaboost, and a procedure for handling the tradeoff between the TPR, the FPR and the number of features in each layer.

3 A learning framework based on nested cascades of boosted classifiers

In this section the proposed learning framework (Sect. 3.1), with an especial emphasis in the description of the training procedures (Sect. 3.2), is presented. In addition, a description of a face detection system (Sect. 3.3) and an eyes detection system (Sect. 3.4) built using this framework is given.

3.1 Nested cascade classifier architecture

A nested cascade of boosted classifiers is composed by several integrated (nested) layers, each one containing a boosted classifier. The whole cascade works as a single classifier that integrates the classifiers of every layer. Figure 1 shows the structure of a nested classifier. 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 (strong) classifiers H_C^k. It should be noted that a given classifier corresponds to the nesting (combination) of the previous classifiers. The computation of H_C^k makes use of the already evaluated classifiers:

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

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

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

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

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

$$O_C(x) = \begin{cases} \text{sign}(H_C^q(x)) \\ \quad H_C^k(x) \geq 0, k = 1, \ldots, q-1 \wedge (H_c^q(x) < 0 \vee q = M) \\ \text{sign}(H_C^1(x)) \quad H_C^1(x) < 0 \end{cases} \tag{4}$$

with a confidence value of a positive detection given by:

$$conf_C(x) = H_C^q(x) \ni H_C^k(x) \geq 0, \quad k = 1, \ldots, M \tag{5}$$

3.1.1 Weak classifiers design

The weak classifiers are applied over features computed in every pattern to be processed. Each weak classifier has associated a single feature. The weak classifiers are designed after the *domain-partitioning weak hypotheses* paradigm [21]. Under this paradigm the weak classifiers make their predictions based on a partitioning of the domain X (decision stumps are the simplest example of domain-partitioning classifiers). Each classifier has a value associated with a partition of this domain. The domain is partitioned into disjoint blocks $X_1, \ldots, X_n$, 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 the 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 $F_1, \ldots, F_n$, and a weak classifier h will have an output for each partition block of its associated feature f: $h(f(x)) = c_j \ni f(x) \in F_j$.

For each classifier, the value associated to each partition block (c_j), i.e., its output, is calculated for minimizing a loss function of the margin [21], which is also a bound of the training error. This value depends on (i) the number of times that the corresponding feature—computed on the training samples—falls into this partition block (weighted histograms), (ii) the class of these samples (y_i), and (iii) their importance $D(i)$. The value of c_j is given by [21]:

$$c_j = \frac{1}{2} \ln \left(\frac{W_{+1}^j + \varepsilon}{W_{-1}^j + \varepsilon} \right) \text{ with}$$
$$W_l^j = \sum_{i: f(x_i) \in F_j \wedge y_i = l} D(i) = \Pr\left[f(x_i) \in F_j \wedge y_i = l \right],$$
$$\text{where } l = \pm 1, \tag{6}$$

and ε a regularization parameter.

The outputs of a weak classifier (c_j), obtained during training, are stored in a LUT for speeding up its evaluation. Thus, a weak classifier will be defined by $h(f(x)) = h_{LUT}[x] = LUT[index\ f(x)]$, with $index$ a function that returns the index (block) associated to the feature value $f(x)$ in the LUT. After having evaluated the feature value, when using equally sized partitions and independently of the number of blocks, this implementation allows to compute the weak classifiers in constant time (the same time that takes to evaluate decision stumps). In this case the training time of (6) grows linearly with the number of training examples. Another important advantage of using domain partitioning is the possibility of using features that can not be handled by decision stumps (e.g., mLBP).

3.1.2 Features

The features we employ are rectangular Haar-like features [31] and mLBP features [6]. Rectangular features are simple features that can be evaluated very quickly, independently of their size, using the integral image [31]. We partition the range of each feature in blocks of equal size for obtaining the weak classifiers. mLBPs correspond to a variant of Local binary patterns—LBP (also known as texture numbers or census transform), and unlike rectangular features they are invariant to linear constrast changes. A mLBP feature is obtained by taking a patch of 3×3 pixels within the processing window and comparing each pixel in the patch to the average pixel value of the patch. The output of each of these comparisons is encoded as a 0 or as a 1, and then concate-

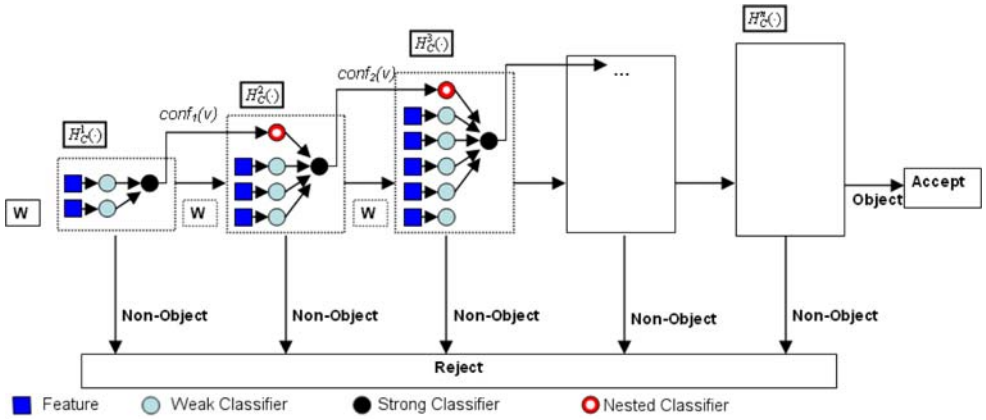

Fig. 1 Nested cascade classifier

nated for obtaining a binary number of 9 bits. Therefore for mLBP features the partition of the domain is already defined by the feature itself.

3.2 Training procedure

When developing a complex learning machine 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 the order in which they are "shown" to the learning machine. It is also important how to train each part of the learning machine. In the case of a nested cascade of boosted classifiers, it is not obvious which the best training strategy is, because several interrelated layers are trained at different moments. The following questions should be answered: Which examples should be presented to a specific part of the machine in a given moment? Which criterion should be used for stopping the training of a given part of the machine? How are these criteria related with the required final performance of the machine (TPR, FPR, and processing speed)?

3.2.1 Design of the strong classifier

The real Adaboost learning algorithm [21] is employed for selecting the features and training the weak classifiers $h_t^k(x)$. The implemented real Adaboost learning algorithm takes into account both, the nested configuration of the cascade [a given layer of the cascade should not be trained without taking into account the previously trained layers, see (2)–(4)], and also the asymmetrical distribution of the two classes. The pseudo code of this algorithm is shown in Fig. 2.

The main idea of cascade classifiers is to process most non-object windows as fast as possible, and to process carefully the object windows and the object-like windows. The final TPR and FPR of the whole classifier are the product of the corresponding rates in each layer. Therefore, for obtaining a high TPR and low FPR for the whole cascade classifier, high TPRs (larger than 0.99) but reasonable low FPRs (lower than 0.5) in each layer are required. For instance, a cascade with ten layers and TPR and FPR values of 0.999 and 0.2, in each layer, will produce a resulting TPR and FPR of $0.99(0.999^{10})$ and $1.024 \times 10^{-7}(0.20^{10})$, respectively. Therefore the design and selection of the optimal parameters of each node of the cascade is very important. For handling the tradeoff between the TPR, the FPR and the processing speed we do not fix the number of features nor the target rates in each layer as most of the works do [6,31,32,35]. We manage this by setting the maximum allowed FPR (*fprMax*) and the minimum allowed TPR (*tprMin*) per layer, while the minimum number of features is selected such that *fprMax* and *tprMin* are achieved. At each iteration of Adaboost several values of a bias, $b_k > 0$, are tested for fulfilling the classification requirements of the layer (see Fig. 2, *ValidateClassifier* function).

As [25] we need to select a priori two parameters. The main difference between our procedure and [25] is that we minimize the number of features following a greedy procedure that assures the minimum desired TPR for each layer, while Sun et al. [25] propose a greedy optimization procedure for maximizing the TPR using a cost function that considers the target FPR and the expected number of evaluated features.

3.2.2 Selection of the training examples

How to select adequate negative examples when training a specific part of a nested cascade of boosted classifiers is not obvious. Moreover, detection problems require discriminative analysis between objects and non-objects (the rest of the world). This produces two types of asymmetries: (1) there is a high asymmetry in the a priori probability of occurrence of the classes: in an image there are much more non-object windows than object windows, and (2) one of the classes is the negation of the other, therefore there is a high asymmetry in the *"size"* of the classes (as regions of the input space).

Fig. 2 Real Adaboost training algorithm for a layer k of a nested cascade

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

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

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

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

normalize D_1 **to a p.d.f.**

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

$t \leftarrow 1$

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

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

// train all weak classifiers :

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

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

$$Z_p = 2\sum_j \sqrt{W_{+1}^j W_{-1}^j}$$

}

select $h_t, t = \arg\min_{p=1,...,P} Z_p$

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

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

normalize D_{t+1} **to a p.d.f.**

$t \leftarrow t + 1$

}

}

ValidateClassifier$(H_C^k, k, PV, NV, frpMax, tprMin)$\{

for b_l **in** $B, l = 1,..., L$ **do**

$\overline{H} \leftarrow H_C^k - b_l$

if $(fpr(\overline{H}, NV) \leq \prod_{l=1}^{k} frpMax \wedge tpr(\overline{H}, PV) \geq \prod_{l=1}^{k} tprMin)$\{

$H_C^k \leftarrow \overline{H}$

return TRUE

}

return FALSE

}

with:

PT / PV : Positive Training/Validation Set ; NT / NV : Negative Training/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

From several millions examples (the rest of the world), one should select the ones that correctly define the classification boundary (positive class versus negative class). If we take a face detection system as a case study, every window of any size in any image that does not contain a face is a valid non-object training example. Obviously, to include all possible non-face patterns in the training database is not an alternative. For defining such a boundary, non-face patterns that look similar to faces should be selected. This is commonly solved using the bootstrap procedure [26], which corresponds to iteratively train the classifier, each time increasing the negative training set by adding examples of the negative class that were incorrectly classified. When training a cascade classifier the bootstrap can be applied in two different situations: before starting the training of a new layer and for re-training a layer that was just trained. After our experience, it is important to use bootstrap in both situations. When we apply this procedure for a layer already trained, we call it *internal bootstrap*, while when we apply the bootstrap before starting the training of a new layer, we call it *external bootstrap*. The external bootstrap is applied just one time for each layer, before starting its training, while the internal bootstrap can be applied several times during the training of the layer. The bootstrap procedure in both cases is the same (see Fig. 3) with only one

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

Fig. 3 Bootstrap procedure

```
NestedCascadeTraining(){
  C ← {φ} //Nested cascade C, initially empty
  H_C^0 ← 0
  k ← 0 // Cascade layers counter
  do{ // Training procedure of cascade layer k
    k ← k + 1
    NT_k ← Bootstrap(C, InitSizeNT, NegIMTrainSet) //External bootstrap
    NV_k ← 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)
      • C̄ ← C ⋃ H_C^k
      • BNT_k ← Bootstrap(C̄, (FinalSizeNT − InitSizeNT) / B, NegIMTrainSet)
      • NT_k ← NT_k ∪ BNT_k
    C ← C̄
  }while(fpr(C, NV_k) > fprMaxC)
  return C
}
```

with:

PT / PV : Positive Training/Validation Set ; NT_k/NV_k : Negative Training/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

Fig. 4 Nested cascade training procedure using internal and external bootstrap

difference, before starting an external bootstrap all negative samples collected for the training of the previous layer are discarded. The use of internal bootstrap in the training of the layers of a cascade classifier is a key point that until now, to our knowledge, has not been carefully analyzed, being just briefly mentioned in [6].

The training procedure of the whole nested cascade is described in Fig. 4. The nested cascade is trained until the target overall FPR is achieved. The training of each 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).

3.2.3 Training time

As we mentioned in Sect. 2 the training time is an important issue. The training time mainly depends on two factors, the time required to train each layer of the cascade using Adaboost, and the time required to perform the bootstrap. The time required for the bootstrap depends mainly on the computational time required to evaluate the cascade, which becomes

larger for the last layers of the cascade. This happens because more windows need to be analyzed for colleting the negative examples, and because these windows are processed by more layers of the cascade. Therefore the only way to reduce the time required for the bootstrap is having a faster cascade, which we achieve thanks to the use of a nested cascade, and fast weak classifiers (such as the domain-partitioning classifiers previously described). Concerning the training of the layers of the cascade, in the work of [32] the training time for one layer is $O(m\log(m)|F|T)$ plus the time required for the bootstrap, with m the number of training examples of that layer, $|F|$ the number of features being tested, and T the number of selected features in the layer. The $O(m\log(m))$ factor comes from the selection of the threshold of the decision stump. This time can be reduced to $O(m)$ if an "off-line" evaluation of the features for each training example and a sorting of these values is done, but the memory requirement is not longer $O(m)$, but $O(m|F|)$, which could be prohibitive[2] when the training set and number of features are large.

In the present work the training time of the layers is greatly reduced thanks to several factors. First, the weak classifiers can be trained in time $O(m)$ with a memory requirement of $O(1)$, thanks to the use of domain partitioning with equally spaced blocks of the feature domain. This reduces the training time of each layer to $O(m|F|T)$. A second factor for reducing the training time is the sampling of the features before each iteration of Adaboost. This reduces the training time to $O(qm|F|T)$, with $0 < q \leq 1$ the percentage of features considered at each iteration. Third, the training time is further reduced by using rectangular features for the first layers of the cascade and mLBP based features for later layers. For example, when using processing windows of 24×24 pixels and 3×3 pixel neighborhoods for evaluating the mLBP features, there are about 135,000 possible rectangular features and only 484 mLBP features. Therefore the selection of features and classifiers using only mLBP features is about 257 times faster.

3.3 Face detection system

3.3.1 General organization

The block diagram of the face detector system is presented in Fig. 5. The system is based on five main modules. The system is designed for being able to detect faces appearing at different scales and positions within the image being analyzed. First, for detecting faces at different scales, a multiresolution analysis is performed by downscaling the input image by a factor of 1.2 (*Multiresolution Analysis* module). This scaling is performed until the width or the height of the downscaled image is smaller than the processing window (image region) size. Afterwards, in the *Window Extraction* module, windows of 24×24 pixels are extracted for each of the scaled versions of the input image (the cascade classifier was trained for analyzing windows of that size). Depending on the application, not all windows are extracted (see Sect. 3.3.2 for details). The extracted windows can be then pre-processed for obtaining illumination invariance, by using methods like variance normalization [31] or histogram equalization [19]. In our system, thanks to the use of mLBP features we do not perform any kind of preprocessing, which allows a reduction in the processing time. Later on, each of the pre-processed windows is analyzed by the nested cascade of boosted classifiers (*Cascade Classification* module). After all selected windows have been classified as faces or non-faces, in the *Overlapping Detection Processing* module the windows classified as faces are analyzed and fused (normally a face will be detected at different scales and positions) for determining the size and position of the final detections. In this module the confidence values associated to the detections are used for fusing them: if the number of overlapped windows in a given position is larger than a given threshold th_{num}, and also if the *detection volume* [4] of the overlapped face windows in a given position is larger than a threshold th_{vol}, then the face windows 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 [28].

3.3.2 Algorithms flavors

Different flavors of the face detection system designed to be used in different types of applications are implemented. In the following paragraphs a brief description of them is given.

Full search versus *Speed search*: The difference between *Full Search* and *Speed Search* is that in the full search case all possible image's windows are considered, while in the speed search a multi-grid approach is employed for selecting the windows' positions to be evaluated. The speed search is based in the procedure described in [7]. A multi-level grid is defined over the image plane. At the first level of the grid, a step size of 6 pixels is used for selecting the windows' positions. At this resolution almost 2.8% of all possible positions are evaluated. At each analyzed grid position the output given by the cascade, i.e., the confidence of the classification, and the index of the last layer where the window was processed, are fed back to the *Window Extraction* module. At this module it is decided if the image, at neighbor window positions, should be further processed or not. If the confidence is above a threshold *ths1*, a second level of the grid is analyzed, considering a finer grid around each starting point of the coarse grid (using a grid step of 3 pixels). Then, each grid

[2] With current computers' memory this is becoming no longer valid (~4 GB of memory is needed when 10,000 examples and 100,000 features are used).

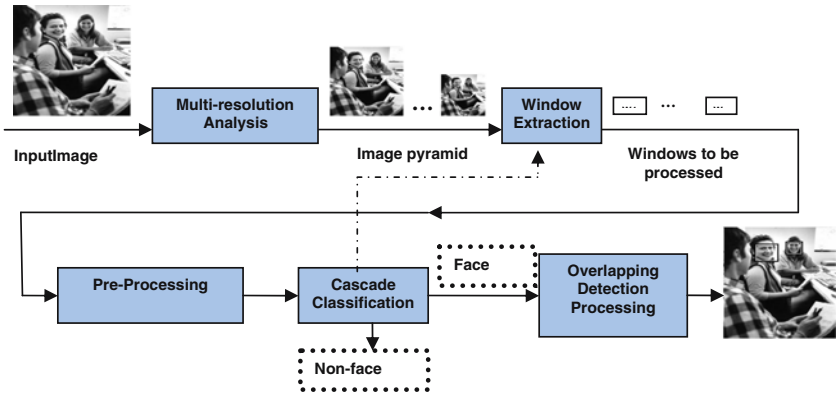

Fig. 5 Block diagram of a face detection system

position with a score value above a threshold *ths2* is evaluated in the third stage using a 3×3 neighborhood. The speed of the process is controlled by the thresholds values *ths1* and *ths2*. Higher threshold's values means higher processing speed, but possible lower detection rates (many true faces can be lost) and false positive rates. Given that a nested cascade is being used, different values of *ths1* and *ths2* for each of the layers have to be selected. This is done using a multi-objective genetic algorithm [29], which optimizes the system for having a fast detection speed, a high TPR and a low FPR.

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 implemented a variant of our algorithm that detects just the most relevant faces in a given image or video frame, which reduces considerably the number of false positives. The *most relevant faces* procedure consists on searching for all faces in a given image, but to filter out the detections 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 for images where only one face is expected.

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).

3.4 Eyes detection system

The eye detector follows the same ideas that the face detector does, i.e., it has its same processing modules. 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 already detected face regions. 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 (the left eye detector in our case), 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 is a boosted classifier consisting of a 1-layer cascade, its weak classifiers are based on rectangular features; it works over windows of 24×24 pixels, and it can process faces of 50×50 pixels or larger. We use only one layer because of two reasons: (1) the bootstrap procedure is not needed, a representative negative training set can be obtained by sampling not-centered windows of eyes, and (2) the processing time is not an important issue because only a small region of the image (the face) needs to be analyzed and the scale of the face is already known. Because the eye detector will be applied to a restricted, reduced image area (upper part of a face), only one flavor of the eye detector is needed, which is equivalent to the *Full-Search* used for the face detector.

4 Experimental methodology and results

In this section we will present a comparative analysis of the critical components of the proposed learning framework. Afterwards we will compare the trained face and eyes detection systems with state of the art similar systems using standard image databases. Finally, we will show the performance of a gender classification system built using the same framework, as an example of the application of this framework to the construction of classification systems.

4.1 Analysis of training the procedure

We have performed an analysis 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 internal-bootstrap, (3) the use of rectangular and/or mLBP features, (4) the effect of the maximum FPR per layer, and (5) the use of feature's sampling.

The whole analysis presented in this section was carried out in 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 family photograph albums, were employed. No single image employed for testing (see Sect. 4.2) was employed in this analysis. The following image datasets were built: PT (Positive Training set): 5,000 training examples obtained from several hundreds images; PV (Positive Validation set): the mirrored version (flop) of all faces from the PT; NIT (Negative Images Training set): 3,500 images not containing faces, used for the bootstrap procedure; NIV (Negative Images Validation set): 1,500 images containing non-faces used for the bootstrap during validation. 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 changes in each layer and in each iteration; and 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 changes for each layer and in each iteration.

The presented analysis was performed using ROC (Receiver Operating Characteristic) curves. In these ROCs, each operation point was obtained by evaluating cascade instances with different number of layers, using the validation sets PV and NV. In other words, the parameter to be changed for obtaining the ROCs is the number of layers of the cascades. In the following experiments the minimum TPR per layer was set to 0.999.

4.1.1 Nested versus non-nested cascades

In Fig. 6a [3] is shown the effect of using a nested cascade versus a non-nested cascade in terms of classification accuracy, while in Fig. 6b 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, the non-nested cascade has larger FPR per layer than the nested one. Moreover, it can be noticed that for the layers 2–4 of the cascade, the non-nested cascade needs more than two times the number of features required by the nested cascade. Given that the training and classification time are directly related with the number of features, a nested cascade has a faster training and operation speed than a non-nested one.

4.1.2 Internal bootstrap

As already explained in Sect. 3.2, we propose to use both internal and external bootstrap. After several experiments we found out that it is better to repeat the internal bootstrap several times, three 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 6(c) shows the effect of performing the internal bootstrap. Clearly the effect for the first layers (the ones with larger FPR) is quite important: the use of internal bootstrap reduces the FPR to the half, while the number of selected features at each layer is almost the same, producing a faster and accurate cascade.

4.1.3 Feature sampling during training

For reducing the training time, at each iteration of the Adaboost feature selection (and weak classifier training) not all features are considered, but only a subset of the possible features [1,28]. We have tested the training algorithm considering 100 and 20% (randomly sampled) of the features at each iteration of Adaboost. In Fig. 6d, are shown the obtained results. As it can be seen, when using less features better results are obtained for the first layers of the cascade, probably because the chance to over fit has been reduced. For the remaining layers of the cascade the performance does not change very much with the number of features. Taking into account this situation, 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 mLBP features. We make three different experiments: training using only mLBP features, training using only rectangular features, and training using both kind of features, rectangular ones for the first two layers and mLBP features for the subsequent layers. As it can be notice in Fig. 6e, the use of only mLBP 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 mLBP features and not just rectangular features? First, mLBP features are invariant to difficult illumination conditions, and they have shown to have a better performance than rectangular features in images with extreme illumination [6]. Second, the number of mLBP features to be selected is much smaller than the number of rectangular features; therefore, the training is much faster when using mLBP features. Taking

[3] Notice that in all graphs shown in Fig. 6, with the exception of Fig. 6b, the operation points shown in the left side (with lower TPR and lower FPR) correspond to later layers of the cascade, while operation points in the right side correspond former layers of the cascade.

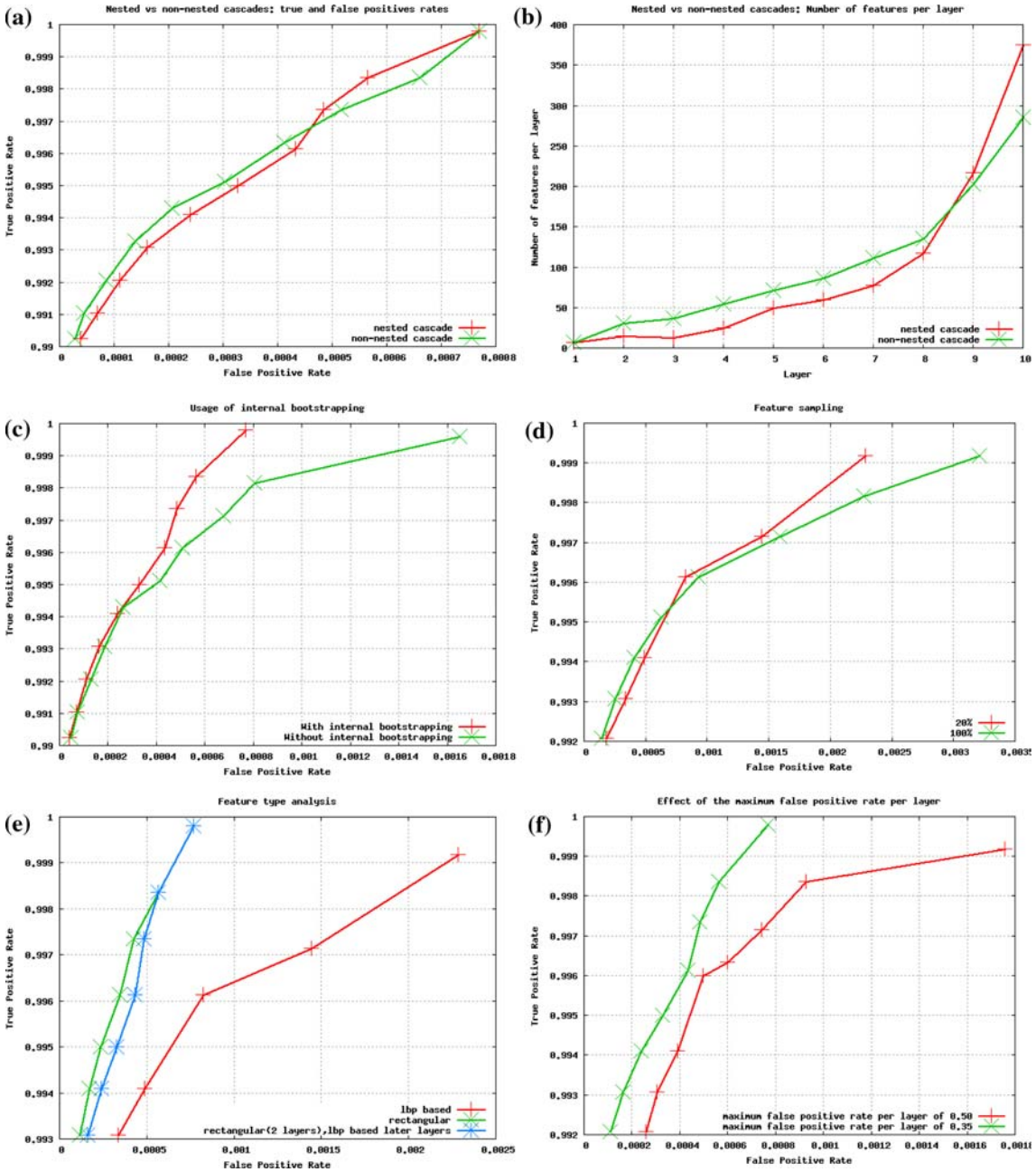

Fig. 6 Analysis of training parameters. **a** Evaluation of nested and non-nested cascade on the validation set. **b** Number of features needed at each stage of the nested and non-nested cascades. **c** Effect of using internal bootstrap. **d** Effect of feature sampling during training. **e** Effect of selecting different features types. **f** Effect of selecting different maximum FPR per layer

this into consideration and also the fact that when using both kinds of features a similar performance is obtained, in our final detection and classification systems we use rectangular features in the first two cascade layers and mLBP 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. As it can be noticed in Fig. 6f, the use of a 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 induce the selection of a few weak classifiers for a layer, which has a negative effects in the following layers. This effect is also seen in some of the experiments done by [21]: at the first iterations of Adaboost the performance is very

poor, for boosted classifiers of sizes from 1 to 5–10, the error even increases when adding new weak classifiers, however after adding at least ten or more classifiers, the error starts to diminish very quickly when new classifiers are added. In our case, when the maximum FPR is large (0.5 in Fig. 6f), only four weak classifiers are needed in the first layer, which is a small number of weak classifier for a boosted classifier. On the other hand, when the maximum FPR is lower (0.35 in Fig. 6f), the number of selected weak classifiers in the first layer is seven. We believe that having a very small number of weak classifiers in the first layers can have an important negative effect in final performance of the cascade classifier.

4.2 Evaluation of the proposed face and eyes detection systems

4.2.1 Experimental datasets

For testing our face and eyes detection 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 (http://www.humanscan.de/support/downloads/facedb.php) consists of 1,521 gray level images with a resolution of 384×286 pixels. 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 illuminations, backgrounds, face sizes, and face expressions. The FERET database [18] 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 [20], we selected 1,016 images containing frontal faces (254 persons, four images for each person) for testing our detection systems. The employed FERET subset is available in http://vision.die.uchile.cl. The CMU-MIT database [19] 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), we do not employ it for evaluating the eye detector. It is important to notice that in some publications people has used different subsets of this dataset, because some of the annotated faces are face drawings; therefore comparison is not always straight forward. In this case we use all 130 images and we considered all 507 faces. The UCHFACE database was especially created for evaluating eyes detection algorithms in images obtained under uncontrolled conditions. It consists of 142 grayscale images obtained from Internet, containing 343 frontal and semi-frontal faces. Face, eyes and gender information was annotated in all these images, and it is available for future studies in http://vision.die.uchile.cl/.

To have a first impression of the capabilities of the built classifiers, in Fig. 7 are presented some selected examples of face and eyes detection, as well as gender classification, in the BioID, FERET, CMU-MIT, and UCHFACE databases.

4.2.2 Face detection evaluation

Our 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 mLBP-based for the subsequent layers. The cascade was trained using a maximum FPR per layer of 0.20 (experimentally this value gave us better classification results and a more compact cascade than when using a maximum FPR per layer equal to 0.35), a minimum TPR per layer of 0.999, three internal bootstraps for the training of each layer, and a positive training set of 5,000 examples. The initial negative training set for each layer had 2,400 examples (obtained using external bootstrap), and in each internal bootstrap steps 400 more examples were added, obtaining a total of 3,600 negative examples for the final training of each layer. The final training time of the whole nested cascade was about 15 h in a 1.8 GHz Pentium 4, with 1,280 MB running Debian GNU/Linux. The obtained final trained cascade has ten layers.

- Face detection in Single Face Images: BioID database. In Fig. 8a are shown the ROC curves of the different face detector flavors on the BioID database. For this database, selected points of these ROCs are shown in the Table 1. In [6] it was 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% [9] and 92.8% [10], but they do not indicate the FPR. In any case these numbers are lower than ours.
- Face detection in Single Face Images: FERET database. In Fig. 8b are shown the ROC curves of the different face detector flavors in FERET. Some selected points of these ROCs are shown in Table 2. No other groups have reported face detection results in this subset of FERET. Nevertheless, 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

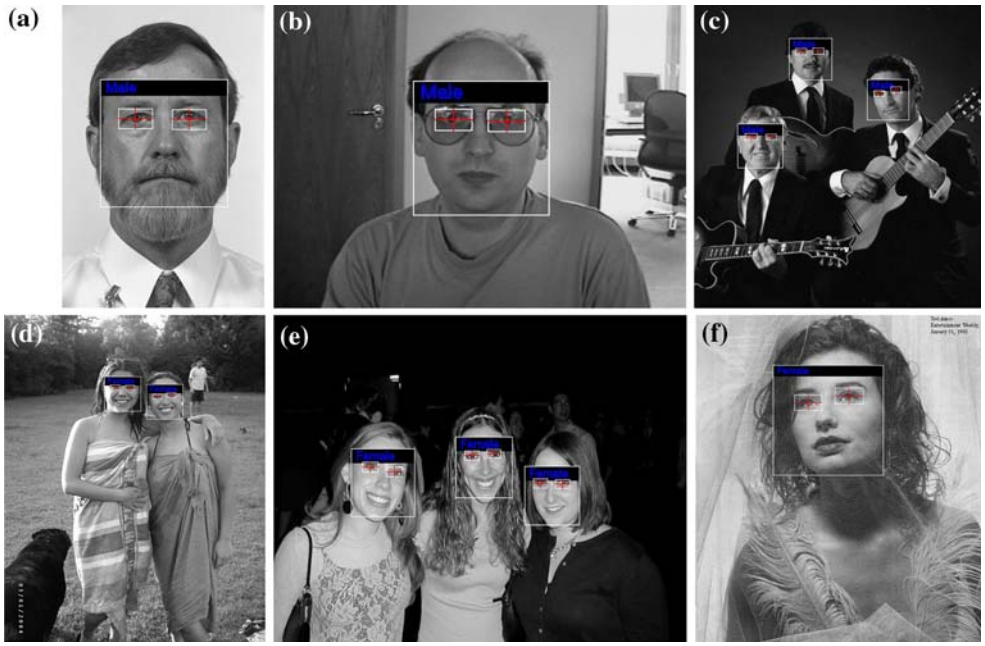

Fig. 7 Some selected examples of our face detection, eyes detection and gender classification systems at work on the FERET **a**, BioID **b**, UCHFACE **d–e** and CMU-MIT **f** databases

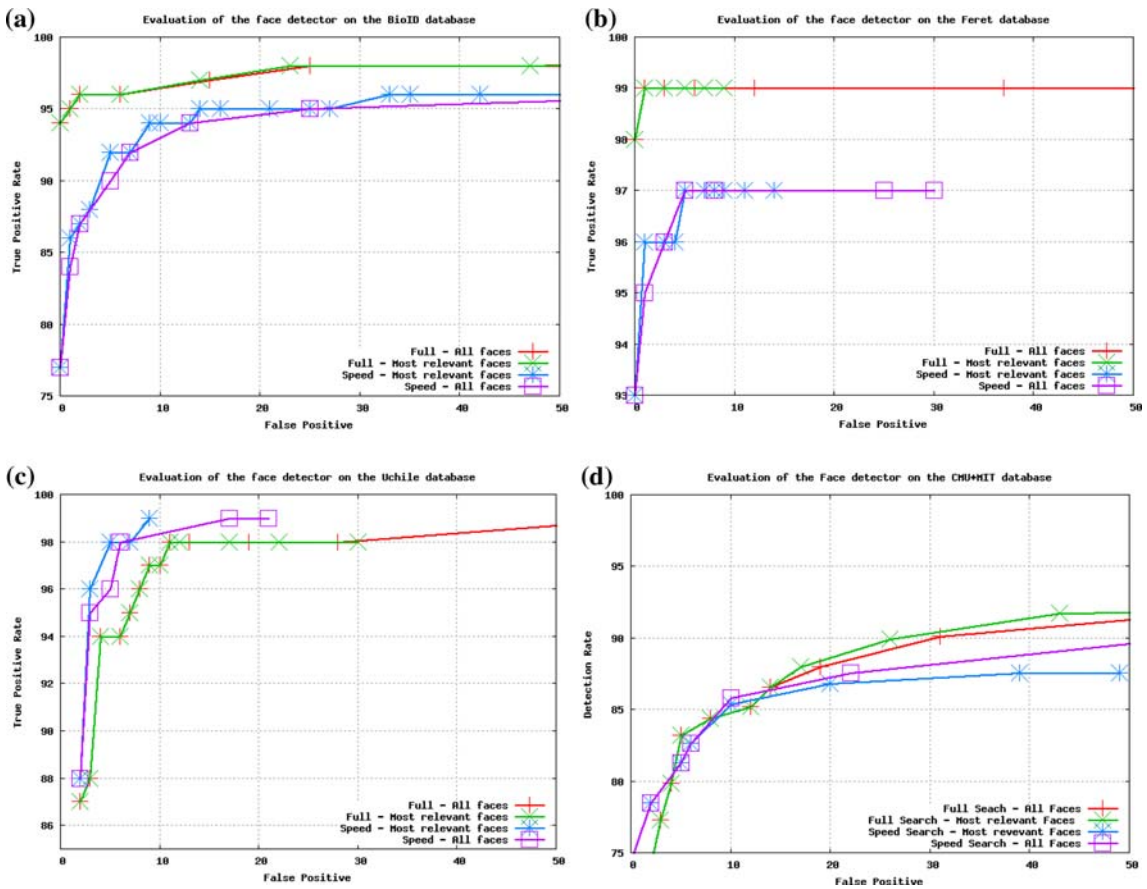

Fig. 8 ROC curves of the different face detector flavors in the BioID **a**, FERET **b**, UCHFACE **c**, and CMU-MIT **d** databases

Table 1 Comparative evaluation (TPR) 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 and Ernst [6]		∼50		∼65				∼84	∼94	∼98

Table 2 Comparative evaluation (TPR) 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	

Table 3 Comparative evaluation (TPR) 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	

system after the face detection stage (FERET is a face recognition test set).

– Face detection in Multiple Face Images: UCHFACE database. In Fig. 8c 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 three false positives, 98.5% with five false positives).
– Face detection in Multiple Face Images: CMU-MIT database. In Fig. 8d 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, all the results here presented were obtained by preprocessing the images with a 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 Fig. 8d all flavors have similar performance for low FP (false positives) values. However, for operation points with larger FP, the *Full-Most* flavor of the face detector gives better results. If we compare these results to the ones obtained by other methodologies presented on the literature in terms of TPR and FP, we obtain better results than [19,31], slightly better results than [12], slightly worse results than [4] (but our system is much faster[4]), and worse results than [35], [22] and [2]. It is difficult to compare our system with [6], because they use a reduced subset of the CMU-MIT databases. But considering that using the complete database is more difficult, because drawings of faces are considered on the original dataset, our results are better than the ones of Fröba and Ernst [6]. We think that we have lower detection rates than Wu et al. [35] and Schneiderman [22] 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 [35] 20,000 training faces are employed. The better performance of Brubaker et al. [2] might be because of two reasons: the use of CARTs (Classification And Regression Trees) as weak classifiers, and the criteria used for selecting the trade-off between the FPR and TPR of the classifier. Although Brubaker et al. [2] does not give any number, we think that the processing time and training time of our detection system is shorter. One of the reason is that we use weak classifiers, which can be evaluated just by reading one array (LUT) value—after evaluating the feature—while Brubaker et al. [2] used CART classifiers of depth four, which require at least four comparison evaluations, therefore is at least four times slower. In terms of the training time, as already mentioned, in our case it takes $O(m|F|T)$ while the training time of Brubaker et al. [2] is at least $O(m\log(m)|F|T)$.

4.2.3 Eyes detection evaluation

As already mentioned, our eye detector was trained using a 1-layer cascade, and the initial training set was not modified using the bootstrap procedure. This difference with the training of the face detector comes from the fact that we aim to detect eyes only within face regions (the negative examples domain is constrained), therefore the bootstrap step

[4] That system is about eight times slower than Viola and Jones [31,32]. Our system has about the same processing speed than Viola and Jones [31,32].

Table 4 Comparative evaluation (TPR) 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 and Ernst [6][a]	∼66		∼87							∼90				
Wu et al. [35]		89			90.1	90.7							94.5	
Viola and Jones [31]					76.1							88.4		92
Rowley et al. [19]					83.2							86		
Schneiderman [22]				89.7				93.1			94.4			
Schneiderman and Kanade [23]														94.4
Li et al. [12]					83.6							90.2		
Brubaker et al. [2]				89.1	90.5							93.1		
Delakis and Garcia [4]	88.8				90.5							91.5		92.3

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

is not needed. The used training set was generated using faces contained in one of our face databases (this database can not be make public). The positive examples were extracted by cropping windows centered at eyes positions, while the negative examples (non-eye examples) were obtained by cropping windows from face area positions that are not centered in the eyes (it should be remembered that the eye detector is always applied over faces windows). The size of the positive training and validation sets was 6,000 in each case, while the size of the negative (non-eye) training and validation sets was 20,000 in each case.

Several eyes detection algorithms have been proposed during the last years. For comparison purposes, we selected state of the art eyes detection algorithms that 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 variable illumination conditions. Two recently proposed systems that fulfill these requirements are [15] and [5]. Both of them make use boosted classifiers, are applied after previous stage of face detection, and have been evaluated in the BioID database.

For evaluating the eyes detection accuracy only correctly detected faces are used. We employ cumulative curves of eyes localization relative error [15]. The relative error is defined as the Euclidian distance between the ground truth of the eyes and the centers of the detected eyes, normalized by the Euclidian distance between the ground truth eyes centers. We have considered as center of the eyes the middle point between the boundaries of the eye. In Fig. 9 are shown these cumulative curves for the eye detector on the BioID, FERET, and UCHFACE databases using the different flavors of the face detector. 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. It can be noticed that all search flavors used for the face detector gives almost the same results for the eye detector. Some selected points of these error cumulative curves, when using the *Full-All* flavor of the face detector, together with the mean error in pixels for the eyes detection are shown in Table 5. The obtained results are very accurate: for example 3.02 pixels error and 97.83% detection rate (DR) in the BioID database. The average distance between the eyes for the BioID database is 54 pixels, hence for a 97.83% detection rate the normalized error (in terms of the eyes distance) is only 5.6%. In the case of the FERET images, for a 99.65% DR the normalized error is 5.38%. In the case of the UCHFACE images, for a 95.16% DR the normalized error is 5.08%.

This small error (close to 5% when all eyes are detected) might be due to an inaccurate annotation of the ground truth (both for the training and evaluation datasets). For example, when annotating a face in which the distance between the eyes is 100 pixels, it is very likely that there will be an error of 5 pixels in the annotation. We think that the obtained accuracy of the detector is very close to one of the human being, and that it would be very difficult to obtain more accurate results without a very careful annotation of the training faces, and a processing performed at larger faces' resolutions.

By looking at the cumulative error curves on the BioID database we observe that we obtain much better results that the ones reported in [15]. For a given eye DR, the error we obtain is less than 50% the one obtained in that work. For instance, for DR of 80% we obtain an error of about 0.047, while in [15] the error is 0.1. In [5] a completely different methodology is employed for evaluating the performance of the eye detector. No curves are given but median accuracy measured in terms of *irisis*. It is very difficult to compare this kind of

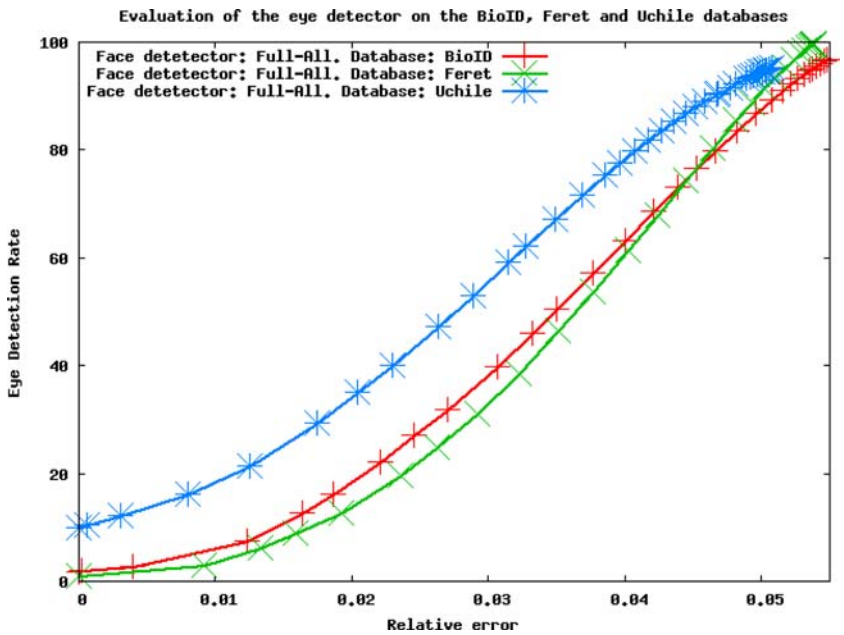

Fig. 9 Cumulative curves of eye localization error of the eye detector on the BioID, FERET, and UCHFACE databases

Table 5 Comparative evaluation of the eyes detection on the BioID, FERET and UCHFACE databases when the Full-All flavor of the face detector is used

BioID								
DR %	31.8	39.8	50.6	57.2	68.7	79.9	89.4	97.8
Normalized error	0.027	0.031	0.035	0.038	0.0421	0.047	0.051	0.056
Mean error in pixels	1.47	1.67	1.90	2.03	2.27	2.51	2.73	3.02
FERET								
DR %	31.0	38.6	53.8	61.3	68.0	80.4	88.8	99.7
Normalized error	0.029	0.032	0.038	0.040	0.042	0.046	0.049	0.0548
Mean error in Pixels	1.84	2.06	2.45	2.65	2.81	3.13	3.35	3.69
UCHFACE								
DR %	29.3	40.0	47.2	59.2	71.7	79.9	90.3	95.2
Normalized error	0.017	0.023	0.026	0.031	0.037	0.041	0.047	0.051
Mean error in pixels	0.69	0.86	0.96	1.12	1.29	1.41	1.61	1.71

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 have reported eyes detection results in 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.

4.3 Gender classification using the proposed framework

Several methods have been proposed for solving the gender classification problem, including neural networks, PCA projections, SVM classifiers, and Adaboost classifiers. Best reported results have been obtained using SVM [3,16] and Adaboost [24,34]. We will briefly analyze some relevant works. In [24] a gender classification system based on Adaboost, that uses decision stumps and rectangular features, is presented. The system reached a performance of 79% correct rate in a set of face images obtained from Internet that were manually annotated and cropped prior to the classification. On the same dataset, this system was favorably compared against the one proposed in [16], being 1,000 times faster and having a higher classification rate (79% against 75.5%). In [34] is described a LUT-based Adaboost system for 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 of AAM (Active Appearance Model). After alignment, gray-level normalization (histogram equalization) is performed. The system achieves a classification rate of 88% on images downloaded form Internet (using 36×36 face windows), and it is favorably compared against a SVM-based system and a decition-stumps based Adaboost system.

We have applied the proposed framework to gender classification using facial information. The implemented gender classification system is applied after face and eyes detection. Face detection is employed for obtaining face windows,

Table 6 Gender classification. Correct classification rates at operation points with equal error rates in both classes

Database	SVM (RBF)	Adaboost.-Rectangular	Adaboost.-mLBP
UCHFACE	79.82	79.22	**80.12**
FERET	84.13	83.95	**85.89**
BioID	79.05	79.52	**81.46**

Only best performing methods are shown. Faces were cropped using automatically detected eyes. Best results are shown in bold

Table 7 Average gender classification processing time on a given face image

Method	SVM	PCA	SVM+PCA	Adaboost-Rectangular	Adaboost-mLBP
Time (ms)	10.48	625	205	**0.244**	1.465

This time does not include the time required for the face detection, face scaling and eyes detection

which are aligned using the detected eyes and then downscaled to 24×24 window's size. The gender classifier is built using Adaboost classifiers. Implemented features are rectangular and mLBP. Using these features, different flavors of the gender classification system were built. These flavors were evaluated using the FERET, BioID and UCHFACE databases, and compared against SVM-based systems (see details in [30]). As can be observed in Table 6, when using mLBP features, the proposed gender classification system outperforms SVM-based systems and Adaboost with rectangular features in terms of classification rate. In Table 7 it is shown the average time required by the different methods for the gender classification of a given face image. It can be seen that Adaboost-mLBP is about ten times faster than SVM-based systems, while Adaboost-Rectangular is six times faster than Adaboost-mLBP, and 60 times faster than SVM. To compare with other methods is not easy because the previously used databases are not available. Nevertheless we can observe that the obtained results are similar to the ones obtained by Shakhnarovich et al. [24] and slightly lower than Wu et al. [34]. Notice that Shakhnarovich et al. [24] does not automatically detect the face and eyes, and that we can handle smaller faces than Wu et al. [34].

5 Conclusions

An important goal of machine vision is to develop systems that can detect objects in cluttered backgrounds, with the ability of generalization across intra-class variability. In this context, in the present work we have described a unified learning framework for object detection and classification using nested cascades of boosted classifiers. The most interesting aspect of this framework is the integration of powerful learning capabilities together with effective training procedures. This framework is based on the use of Adaboost for the training of a nested cascade classifier, domain-partitioning for the design of weak classifiers, a procedure for handling the tradeoff between the classification rates and the complexity of each layer, and rectangular and mLBP features. For the training of the detection systems, internal and external bootstrap is used, which together with feature sampling during training, combined use of rectangular features in the first two cascade layers, and mLBP features in the subsequent layers, and an adequate selection of the maximum FPR per layer allows us to train a face detection system in about 15 h using in a Pentium 4 personal computer.

The framework has been used for the development of face detection and eyes detection systems. We have compared both systems with state of the art similar systems. Our face detection system obtains the best-reported results on the BioID database, and the best reported results on the CMU-MIT database. Our eyes detection system obtains an important improvement over the best-reported results on the BioID database. For future comparisons with the here presented systems, a performance evaluation was carried out in images of the FERET database and in 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.

We have also used the proposed framework for building gender classification systems that were evaluated using the FERET, BioID and UCHFACE databases. The main improvements over previous works are: (1) the use of more suitable features for addressing this problem—mLBP features behave better than rectangular features; (2) the usage of smaller face windows (24×24) which allows analyzing smaller faces, and (3) a faster processing, because, besides the eye alignment, we do not perform any geometric or photometric normalization. From the shown experiments it can be concluded that these systems achieve high accuracy in dynamical environments, and that they largely outperform SVM-based systems in terms of processing speed.

As has been shown, the most interesting aspect of this framework is the possibility of building detection and classification systems with high accuracy, robustness, high processing speed, and high training speed. In a near future we plan to use the proposed learning framework for building other classification tools for the analysis of faces (race classification, face expressions detection, mouth detection, etc.), and for the detection of other kinds of objects. We plant to extend the framework to detect objects 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.

Acknowledgments This research was funded by Millenium Nucleus Center for Web Research, Grant P04-067-F, Chile. Part of the research in this paper use the FERET database of facial images collected under the FERET program. During part of this research work the authors took part of the Alfa project N°AML//19.0902/97/06660/II-0366-FA and they would like to acknowledge its support.

References

1. Baluja, S., Sahami, M., Rowley, H.A.: Efficient face orientation discrimination. International Conference on Image Processing **1**, 589–592 (2004)
2. Brubaker, S.C., Mullin, M.D., Rehg, J.M.: Towards optimal training of cascaded detectors. ECCV 2006: 9th European conference on computer vision, Graz, Austria, 7–13 May, 2006, Proceedings, vol. 1, pp. 325–337, Lecture Notes in Computer Science 3951 (2006)
3. Buchala, S., Davey, N., Frank, R.J., Gale, T.M., Loomes M., Kanargard, W.: Gender classification of face images: the role of global and feature-based information. ICONIP 2004, Calcutta, India, Lecture Notes in Computer Science 3316, pp. 763–768 (2004)
4. Delakis, M., Garcia, C.: Convolutional face finder: a neural architecture for fast and robust face detection. IEEE Trans. Pattern Anal. Mach. Intell. **26**(11), 1408–1423 (2004)
5. Fasel, I., Fortenberry, B., Movellan, J.: A generative framework for real time object detection and classification. Comput. Vision Image Understand. **98**, 182–210 (2005)
6. Fröba, B., Ernst, A.: Face detection with the modified census transform. Sixth Int. Conf. on Face and Gesture Recognition, pp. 91–96 (2004)
7. Fröba, B., Küblbeck, Ch.: Robust face detection at video frame rate based on edge orientation features. Fifth Int. Conf. on Automatic Face and Gesture Recognition, pp. 342–347 (2002)
8. Hjelmås, E., Low, B.K.: Face detection: a survey. Comput. Vision Image Understand. **83**, 236–274 (2001)
9. Jesorsky, O., Kirchberg, K.J., Frischholz, R.W.: Robust face detection using the Hausdorff distance. Third Int. Conf. on Audio- and Video-based Biometric Person Authentication, Lecture Notes in Computer Science, LNCS-2091, pp. 90–95 (2001)
10. Kirchberg, K.J., Jesorsky, O., Frischholz, R.W.: 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 (2002)
11. Kölsch, M., Turk, M.: Robust Hand Detection. Sixth Int. Conf. on Face and Gesture Recognition, pp. 614–619 (2004)
12. Li, S.Z., Zhu, L., Zhang, Z.Q., Blake, A., Zhang, H.J., Shum, H.: Statistical learning of multi-view face detection. Seventh Eu. Conf. on Computer Vision, Lecture Notes in Computer Science, vol. 2353, pp. 67–81 (2002)
13. Lin, Y.-Y., Liu, T.-L., Fuh, C.-S.: Fast object detection with occlusion. Eighth European Conf. on Computer Vision, Lecture Notes in Computer Science 3021, pp. 402–413 (2004)
14. Liu, C., Shum, H.-Y.: Kullback-leibler boosting. IEEE Conference of Computer Vision and Pattern Recognition, pp. 587–594 (2003)
15. Ma, Y., Ding, X., Wang, Z., Wang, N.: Robust precise eye location under probabilistic framework. Sixth Int. Conf. on Face and Gesture Recognition, pp. 339–344 (2004)
16. Moghaddam, B., Yang, M.-H.: Learning gender with support faces. IEEE Trans. Pattern Anal. Mach. Intell. **24**(5), 707–711 (2002)
17. Osuna, E., Freund, R., Girosi, F.: Training support vector machines: an application to face detection. IEEE Conference of Computer Vision and Pattern Recognition, pp. 130–136 (1997)
18. Phillips, P.J., Wechsler, H., Huang, J., Rauss, P.: The FERET database and evaluation procedure for face recognition algorithms. Image Vision Comput. J. **16**(5), 295–306 (1998)
19. Rowley, H., Baluja, S., Kanade, T.: Neural network-based detection. IEEE Trans. Pattern Anal. Mach. Intell. **20**(1), 23–28 (1998)
20. Ruiz-del-Solar, J., Navarrete, P.: Eigenspace-based face recognition: a comparative study of different approaches. IEEE Trans. Syst. Man Cybernetics C (Special Issue on Biometric Systems) **35**(3), 315–325 (2005)
21. Schapire, R., Singer, Y.: Improved boosting algorithms using confidence-rated predictions. Mach. Learn. **37**(3), 297–336 (1999)
22. Schneiderman, H.: Feature-centric evaluation for efficient cascade object detection. IEEE Conference of Computer Vision and Pattern Recognition, pp. 29–36 (2004)
23. Schneiderman, H., Kanade, T.: A statistical model for 3D object detection applied to faces and cars. IEEE Conf. Comput. Vision and Pattern Recogn. **1**, 746–751 (2000)
24. Shakhnarovich, G., Viola, P., Moghaddam, B.: A unified learning framework for real time face detection & classification. Int Conf. on Automatic Face & Gesture Recognition, FG 2002, pp. 16–26 (2002)
25. Sun, J., Rehg, J.M, Bobick, A.F.: Automatic cascade training with perturbation bias. IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2004), vol. 2, pp. 276–283 (2004)
26. Sung, K., Poggio, T.: Example-based learning for viewed-based human face deteccion. IEEE Trans. Pattern Anal. Mach. Intell. **20**(1), 39–51 (1998)
27. Torralba, A., Murphy, K., Freeman, W.: Sharing visual features for multiclass and multiview object detection. AI Memo 2004-008, MIT, CSAIL Laboratory (2004)
28. Verschae, R., Ruiz-del-Solar, J.: A hybrid face detector based on an asymmetrical adaboost cascade detector and a wavelet-Bayesian-detector. Lecture Notes in Computer Science 2686, Springer, Heidelberg, pp. 742–749 (2003)
29. Verschae, R., Ruiz-del-Solar, J., Köppen, M., Garcia, R.V.: Improvement of a face detection system by evolutionary multi-objective optimization. Fifth Int. Conf. on Hybrid Intelligent Systems, pp. 361–366 (2005)
30. Verschae, R., Ruiz-del-Solar, J., Correa, M.: Gender classification of faces using adaboost. CIARP 2006. Lecture Notes in Computer Science. vol. **4225**, 68–78 (2006)
31. Viola, P., Jones, M.: Rapid object detection using a boosted cascade of simple features. IEEE Conf. on Computer Vision and Pattern Recognition, pp. 511–518 (2001)
32. Viola, P., Jones, M.: Fast and robust classification using asymmetric adaboost and a detector cascade. Advances in Neural Information Processing System 14.. MIT Press, Cambridge (2002)
33. Viola, P., Jones, M., Snow, D.: Detecting pedestrians using patterns of motion and appearance. Nineth IEEE Int. Conf. on Computer Vision **2**, 734–741 (2003)
34. Wu, B., Ai, H., Huang, C.: LUT-based Adaboost for gender classification. Fourth Int. Conf. on Audio and Video-based Biometric Person Authentication, June 10–11, Guildford, U.K. (2003)
35. Wu, B., Ai, H., Huang, C., Lao, S.: Fast rotation invariant multi-view face detection based on real Adaboost. Sixth Int. Conf. on Face and Gesture Recognition, pp. 79–84 (2004)
36. Wu, J., Mullin, M.D, Rehg, J.M.: Linear asymmetric classifier for cascade detectors. Proceedings of the 22nd International Conference (ICML 2005), pp. 988–995 (2005)

37. Wu, J, Rehg , J.M., Mullin, M.D.: Learning a rare event detection cascade by direct feature selection. Proc. Advances in Neural Information Processing Systems 16 (NIPS*2003). MIT Press, Cambridge (2003)
38. Xiao, R., Zhu, L., Zhang, H.-J.: Boosting chain learning for object detection. Nineth IEEE Int. Conf. on Computer Vision **1**, 709–715 (2003)
39. Yang, M., Ahuja, N., Kriegman, D.: Mixtures of linear subspaces for face detection. Fourth IEEE Int. Conf. on Automatic Face and Gesture Recognition, pp. 70–76 (2000)
40. Yang, M.-H., Roth, D., Ahuja, N.: A SNoW-based face detector. Solla, S.A., Leen, T.K., Müller, K.-R. (eds.) Advances in Neural Information Processing Systems 12, pp. 855–861 (2000)
41. Yang, M., Kriegman, D., Ahuja, N.: Detecting faces in images: a survey. IEEE Trans. Pattern Anal. Mach. Intell. **24**(1), 34–58 (2002)

Author biographies

Rodrigo Verschae is currently a Ph.D. candidate at the Universidad de Chile, Chile, taking part of the ALFA program in computer vision at the Center of Mathematics and their Applications (CMLA), École Normale Supérieure (ENS) de Cachan, France, place where he received a M.S. degree in Mathematics, Vision and Learning in 2006. From October 2003 to March 2005 he was in the Department of Security Technologies at Fraunhofer-IPK, Berlin, Germany, first, as a research visitor and later as a research associate. During this period he was a research collaborator of the Center for Web Research of the Universidad de Chile. He received an Electrical Engineering degree, and B.Sc. degrees in Computer Engineering and Electrical Engineering (in 2003, 2002 and 2001, respectively) at the same university. His research interests include machine learning and computer vision applied to image analysis and object detection problems.

Javier Ruiz-del-Solar received his diploma in Electrical Engineering and M.S. degree in Electronic Engineering from the Technical University Federico Santa Maria (Chile) in 1991 and 1992, respectively, and the Doctor–Engineer degree from the Technical University of Berlin in 1997. In 1998 he joined the Electrical Engineering Department of the Universidad de Chile as an Assistant Professor. In 2001, he became the Director of the Electro-Technologies Laboratory and in 2005 Associate Professor. His research interests include Mobile Robotics, Computer Vision, Face Analysis, and the Application of Computational Intelligence Techniques to Pattern Recognition problems. Dr. Ruiz-del-Solar is a recipient of the 2004 RoboCup Engineering Challenge Award and IEEE RAB Achievement Award 2003, and since 2006 he was a senior member of the IEEE.

Mauricio Correa Pérez graduated with Highest Honors as Electrical Engineer at the Universidad de Chile, Santiago, Chile, in 2006. He is a member of the Computational Vision Laboratory at the Department of Electrical Engineering of the Universidad de Chile since 2005. Currently he is a Ph.D. student at the Universidad de Chile and he is following the M.S. in Mathematics, Vision and Learning, taking part of the ALFA program in Computer Vision at the Center of Mathematics and their Applications (CMLA), École Normale Supérieure (ENS) de Cachan, Cachan, France. His research interests include computer vision, machine learning, pattern recognition, image and video processing, Robotics and Autonomous Systems.