ON IMPROVING FACE DETECTION PERFORMANCE BY MODELLING CONTEXTUAL INFORMATION

Cosmin Atanasoaei Chris McCool
Sébastien Marcel

Idiap-RR-43-2010

DECEMBER 2010

On Improving Face Detection Performance by Modelling Contextual Information

Cosmin Atanasoaei, Chris McCool, *Member, IEEE,* and Sébastien Marcel

***Abstract*—In this paper we present a new method to enhance object detection by removing false alarms and merging multiple detections in a principled way with few parameters. The method models the output of an object classifier which we consider as the *context*. A hierarchical model is built using the detection distribution around a target sub-window to discriminate between false alarms and true detections. Next the context is used to iteratively refine the detections. Finally the detections are clustered using the Adaptive Mean Shift algorithm.**

The specific case of face detection is chosen for this work as it is a mature field of research. We report better results than several baseline methods on the MIT+CMU, WEB and CINEMA face databases. We significantly reduce the number of false alarms while keeping the detection rate at approximately the same level and in certain conditions we recover misaligned detections.

***Index Terms*—Context Modelling, Face Detection, Multiple Detections.**

I. Introduction

A variety of applications like video surveillance, biometric recognition and human-machine interface systems depend on robust face detection algorithms. In the last decade there has been an increasing interest in real-time systems with high accuracy and many successful methods have been proposed. Despite this, face detection remains a challenging problem and there are improvements to be made.

The task of face detection is a specific case of object detection. Object detection can be described as the task of finding all instances of an object (for instance the face) in an image. Research to date has dealt mainly with the issue of building a robust and accurate object classifier. An object classifier tells if an object is found at a specific position and scale (referred to as a sub-window) in an image. For instance work by Froba et al. [1] and Viola and Jones [2] has provided significantly improved face classifiers, more details are provided in Section II-A.

There are many ways to obtain the sub-windows from an image. A sliding window approach [3] is usually used to find all the object instances in the image often referred as scanning the image (see Section II-B). This can result in multiple detections and false alarms as shown in Fig. 1. A merging and pruning heuristic algorithm is then typically used to output the final detections (see Section II-C).

Recent work has been done to overcome the limitations of the sliding window approach by using a branch-and-bound technique to evaluate all possible sub-windows in an efficient way [4]. The authors build a model that also predicts the location of the object [5]. However, it is not clear how to use this method for different classifiers or feature types (because of the upper bound needed by the branch-and-bound algorithm) or to detect multiple objects.

We propose a more principled method with less parameters. We learn a model that uses the output of the object classifier which we consider as the *context* to distinguish between false alarms and true detections (see Section IV). Our approach is inspired by the work of [6] and [7], but there are many significant differences. These are: i) we use more features than just the model score, computed on various location and scale axis combinations to better describe the detection distribution and we propose a different model following a more discriminative approach, ii) we use this contextual information to iteratively refine the object detections which leads to significantly more accurate detections (see Section V-A), and iii) we cluster the refined detections using the Adaptive Mean Shift algorithm described in Section V-B. The experimental procedure is described in Section VI and the results obtained on several face databases and using a popular face classifier are presented in Section VII. Conclusions and future work directions are given in Section VIII.

II. Background literature

This section presents a short review of the two usual components of a face detection system: a face classifier and a scanning procedure. First we present some of the most well known face classifiers trained to verify if a particular sub-window is a face. Next we show how a typical scanning procedure is used to detect faces in images in conjunction with such a face classifier. Finally we expose some of the problems that may appear such as multiple detections and false alarms and we present some methods to overcome them.

A. Face classifiers

Most of the state of the art face detection methods are based on a cascade of boosted classifiers that provide real-time performance with high accuracy. They combine weak classifiers that are slightly better than random using a linear method with weights proportional to their accuracy [8]–[10]. Usually the weak classifiers are associated to a specific feature. The boosting process can be seen as a feature selection method that uses the selected features to build a strong classifier with arbitrary accuracy. Different approaches are based on Neural networks such as the pioneering work from Rowley et al. [3], but also [11] and [12].

The first real-time face detector was developed by Viola and Jones [2]. Its speed was achieved by a cascade of increasingly

C. Atanasoaei, C. McCool and S. Marcel are with the Idiap Research Institute.

complex boosted classifiers designed to reject at the very first stages most of the negative samples. In this cascade the weak classifiers were based on Haar-like features that were computed very efficiently at any position and scale using an integral image. Each stage of the cascade was trained using the Adaboost algorithm. This system had the same accuracy as previous state of the art methods, but it was much faster. Following this work rotated Haar-like features were introduced in [13] and three boosting algorithms (Discrete, Real and Gentle Adaboost) were evaluated. The authors have shown experimentally that Gentleboost performs better than the rest with a lower computational complexity.

Recently the Modified Census Transform (MCT) [1] and the Local Binary Patterns (LBP) [14] have become a more popular choice than Haar-like features for face detection. They present several advantages such as: low computational cost at any position and scale and illumination invariance (very useful in uncontrolled environments).

Different boosting algorithms were proposed to reduce the number of weak classifiers needed to achieve better accuracy than Adaboost. The FloatBoost learning algorithm [15], for instance, backtracks and deletes the weak classifiers that are ineffective. A novel cascade learning algorithm is also proposed in [16] that is based on forward feature selection. It achieves two orders of magnitude faster training time than Viola-Jones approach and yields a classifier of equivalent quality. Chang Huang et al. [17] proposed a nested cascade detector in which the confidence of the strong classifier from the previous layer is used as input along with other weak classifiers to the next layer. This reduces the number of layers and the number of features used to reach the same detection and false alarm rate.

B. Object detection

To detect objects one usually proceeds by *scanning* the image at different positions and scales. At each position and scale a sub-window is formed and tested against a classifier previously trained with geometric normalized samples of size $S_c = (W_c, H_c)$. This is often referred to as a *sliding window approach*.

There are two main sliding window (scanning) methods: the multiscale and the pyramid [18]. The *multiscale* approach varies the size of the scanning sub-window and the classifier has to interpolate its content to S_c in order to decide if the sub-window contains the object or not. The *pyramid* approach computes a set of scaled images, from the original image, and for each one varies the position of a S_c fixed size sub-window. No interpolation is needed for this approach (the sub-window and the classifier's input dimension have the same size) but the image pyramid must be computed first. It can be shown that both methods test the same number of sub-windows and experimental results have shown that they produce similar results.

The choice of sliding window approach is dependent on the feature type or the classifier used. For example the MCT [1] or Haar-like [2] features can be computed very efficiently at any position and scale using integral images and this means that the multiscale approach is more efficient in terms of speed. However, the Neural Network based face classifier proposed by Rowley et al. [3] was evaluated using the pyramid approach. A modified version of the pyramid approach is presented in [19], where the authors use a coarse to fine grid search for each scale, thus refining the search in the areas where a detection had occur at a coarser grid.

For the sliding window approaches the total number of sub-windows to classify is quadratic to the number of pixels. This is because every position in the image must be matched at every possible scale. Therefore, there can be billions of sub-windows even for small images and thus it is inefficient to do an exhaustive search. Typically one uses some heuristics that reduce this number to practical values by limiting the number of scales or searching every N pixels (by having an offset of N pixels between subsequent sub-windows) [2]. Real-time performance can be achieved by limiting the number of sub-windows to process but at the cost of missing some objects.

Usually applying the sliding window approach with any face detector will result in multiple detections and false alarms (see Fig. 1). These detections must be further processed. This is often referred to as *pruning* false alarms and *merging* multiple detections [18].

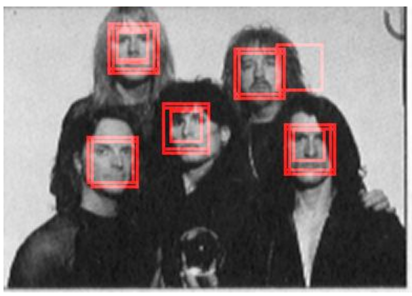


Fig. 1. Typical face detections using the multiscale approach and the MCT boosting cascade classifier described in [1] (without clustering multiple detections and removing false alarms).

Since the true location and the number of objects is not known it is preferred to have a finer scanning than to miss any object or miss predict its location too much. However this will also increase the number of false alarms because a larger number of sub-windows has to be explored. This is because even a state of the art classifier has a false alarm rate (FAR), also called false acceptance rate, that is not zero, usually of the order of 0.1% to achieve good performance. Another reason is that the sliding window approaches test multiple sub-windows that significantly overlap with the true face. This implies that the face classifier fires many times around the true face which results in multiple detections. A coarser scanning will reduce their number but at the cost of missing some faces and producing less accurate detections.

Note that the object classifiers are usually trained only with samples that completely show the face or not at all. During testing a significant number of sub-windows will overlap differently with the true object, for which case there is no guarantee of the classifier output [5]. Still most classifiers will correctly recognize the objects even in these situations.

C. Multiple detections

The most common approach to solve the multiple detection problem is to heuristically merge the ones that form clusters based on the overlapping percentage. Below we present the heuristic methods that have been presented in the literature.

In [2] detections are partitioned into disjoint subsets, by associating two detections to the same one if they overlap. The final step consists of composing for each subset just one sub-window having the average coordinates of all sub-windows in that subset. Some restrictions can be further imposed on the partitioning: two detections are considered in the same subset if they overlap more than a threshold (typically 60% of the area of the biggest detection) and a detection is removed if it is contained by another one. The selection of the best detections (one per subset typically) is done iteratively. At each step the most overlapping detections are merged (typically using non-maxima suppression, averaging or confidence weighting) and then the subsets are computed again. Some variations are implemented in face detection libraries like OpenCV [1] and Torch3vision [2]. Our experiments (see Section VII-B) have shown that this method is sensitive to the scanning parameters. When too coarse, the true detections can be isolated and considered as false alarms and when too fine, false alarms appear in clusters and are usually considered to be a detected face.

A similar heuristic approach was proposed by Rowley et al. [3]. They preserve the detections with higher number of overlapping detections within a small neighbourhood and eliminate the other ones. The final output is given by the centroids of the preserved detections.

A more principled approach was recently proposed in [6] and [7] where the authors study the score distribution in both location and scale space. Their experimental results have shown that the score distribution is significantly different around a true object location than around a false alarm location, thus making it possible to build a model to better distinguish the false alarms and enhance detection. This approach is motivated by the fact that the object classifier is usually trained with geometric normalized positive samples and it does not process the *context* (area around given samples). Also, some false alarm sub-windows may have a higher score than a true detection nearby and may be selected as the final detections using an heuristic merging technique.

III. Our approach

We propose a new post-processing technique for object detection that makes use of the contextual information. We plan to address some problems of the previous methods such as: multiple detections, false alarms and sensitivity to scanning parameters (it should work equally well for fine or coarse face scanning). Our goal is to improve object detection algorithms in general by addressing the specific problem of face detection and in doing so we make the following contributions:

i) **A context-based model** for pruning false alarms is presented in Section IV. We define the *context* as the detection distribution around a target sub-window, by varying its scale and position and checking it against a classifier. The context is described using multiple features such as its density, the geometric distribution for scale and position axis and some score statistics. Our model automatically selects the best features from the contextual information and optimizes its internal parameters.

ii) **Object detection refinement** is performed using this contextual information to improve the accuracy of detections and to make the job of the merging (clustering) algorithm easier. We argue that we can estimate from the context the direction where an object is more likely to reside and we use this idea in a greedy way in the algorithm presented in Section V-A.

iii) **The Adaptive Mean Shift** (AMS) algorithm is used to solve the problem of clustering multiple detections. We favour it against other methods (e.g. average, score weighting, non-maxima suppression) because it uses practically no parameters, the number of clusters does not need to be known a priori and its properties are theoretically established. More details are provided in Section V-B.

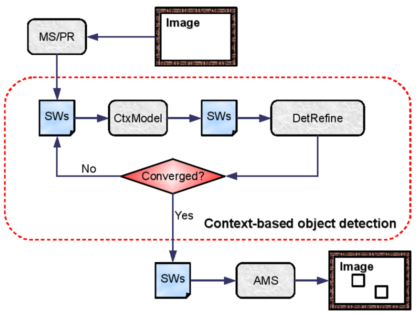


Fig. 2. Context-based face (object) detection method consisting of the following blocks: i) CtxModel - the context-based model to discriminate and remove false alarms (see Section IV-C), initialized with the multiscale or pyramid scanning (MS/PR) and ii) DetRefine - one step of the method to refine detections (see Section V-A). The final block is the AMS clustering algorithm to process the converged collection of sub-windows (SWs) (see Section V-B).

The proposed context-based face detection method is presented in Fig. 2. The first step is to run the multiscale or the pyramid (MS/PR) scanning over the input image using a face (object) classifier. The detections (SWs) are checked against the context-based model (CtxModel) to remove false alarms (see Section IV). These are further refined using the method proposed in Section V-A and a new collection of sub-windows is generated. If the refinement has not converged, the sub-windows are again checked against the context-based model. Otherwise they are clustered using the AMS algorithm presented in Section V-B to merge the detections that were converged closely.

IV. Context-based model

In this section we present a model to discriminate false alarms from true detections. First we describe how we sample around a target sub-window to build its context. Then we present the features we extract from the context and finally the classifier that uses these features to discriminate false alarms.

[1]http://sourceforge.net/projects/opencvlibrary/
[2]http://torch3vision.idiap.ch/

A. Sampling

We sample in the 3D space of location (x, y) and scale (s) to collect detections *around* a target sub-window $T_{sw} = (x, y, s)$. For this we vary its position and scale in all directions (left, right, up, down, smaller and bigger) and we form new sub-windows. Those sub-windows that pass the object classifier are gathered with their associated classifier output ms, referred to as the model score in this paper. The *context of the target sub-window* T_{sw} is defined as the collection of 4D points $C(T_{sw}) = (x_i, y_i, s_i, ms_i)$ obtained as explained.

The goal is to use these points to extract some features from them and to build a classifier that distinguishes between a true object detection and a false alarm. It is important to note that the number of 4D points can vary from one target sub-window to another. For instance a false alarm is supposed to have fewer detections around it than a true object location.

Before we continue, we will define the notations needed to formalize the context sampling. Let N_x, N_y, N_s be the number of points to be considered on each axis (location and scale) along the positive direction. The sampling factors for each axis are defined as dx, dy and ds respectively.

We have used two strategies for context sampling: full and axis. The *full* strategy consists of sampling by varying the location and scale at the same time. First we vary the scale N_s times by multiplying the precedent with $1 + ds$ for larger scales and N_s times by multiplying the precedent with $1 - ds$ for smaller scales. Second we vary, for each scale, the spatial position of the sub-window (left, right, up and down) with constant increments as dx and dy factors of the current width and height. In this case the context can have at most $N_{full} = (2N_x + 1) \times (2N_y + 1) \times (2N_s + 1)$ points. In the *axis* strategy the sampling is done just along one axis at a time. This reduces the maximum size of the context to $N_{axis} = (2N_x + 1) + (2N_y + 1) + (2N_s + 1)$ points.

In our experiments we have used $dx = dy = ds = 0.05$ which corresponds to 5% increments both in scale and position and $N_x = N_y = 6$ and $N_s = 5$ [3]. This makes N_{axis} (at most 37 points) approximately 50 times smaller than N_{full} (at most 1859 points). The axis sampling approach is better suited for real time applications where building the full context may be too expensive.

B. Feature vectors

In the next step we extract a fixed number of low dimensional feature vectors from $C(T_{sw})$. There are two stages for forming the feature vectors: i) the attribute(s) and ii) the axis (and axes) used to obtain the attribute(s).

We use three types of attributes: counts, hits and scores. An overview of the attributes is provided in Table I and a detailed description of each attribute is provided below:

- **Counts** provide a global description of $C(T_{sw})$ by counting detections on some axis combination.
- **Score standard deviation and amplitude** describe the classifier confidence variation across position and/or scale changes. If no detection or just a single detection for the standard deviation case, which happens mostly in the case of the false alarms, the default value is 0.0 that accurately reflects no confidence variation. In the case of true detections this variation is experimentally found to be significantly higher.
- **Hits standard deviation and amplitude** describe the geometrical properties of the context. More precise these features capture, independently for each axis in a combination, the width of the maximum extent where detections still occur. For example for the x-axis case we vary x with N_x positions to the left (indexed as $-N_x$ to -1) and with N_x positions to the right (indexed as 1 to N_x). This gives the detections along x-axis as a collection of indices $\in [-N_x, N_x]$ (which we call *hits*) on which we can compute the amplitude and standard deviation. Up to three dimensional feature vectors are obtained by concatenating such values for all the axis in the combination.

[3]For example, the context for a detection of size 100x100 pixels is obtained by sampling sub-windows from approximately 77x77 to 127x127 pixels and translated by at most 38 pixels.

Attributes	Type	Dimensions
Counts	global	1
Score standard deviation	confidence	1
Score amplitude	confidence	1
Hits standard deviation	geometry	1 - 3
Hits amplitude	geometry	1 - 3

TABLE I
ATTRIBUTES EXTRACTED FROM THE CONTEXT FOR EACH AXIS COMBINATION.

Each feature vector is computed using an attribute and some axis combination (x, y and $scale$ - which gives 7 possible axis combinations). For example we build sub-windows by varying all axes, just two of them (like keeping the $scale$ constant and varying only the x and y sub-window's coordinates) or just one of them (like keeping the x and $scale$ fixed and moving the sub-window up and down).

There are some restrictions when designing these feature vectors. First we need feature vectors that have a meaning even when there is no detection for some axis combination (which implies that no 4D points are generated). This happens especially around false alarms, when the number of detections by varying all axes is well above zero but becomes close to zero when varying just the scale for example. Second, because of the variable number of detections in the contextual area, we want just a single vector of fixed dimensionality to be extracted as a feature.

We have investigated if the above features provide enough information to discriminate between the two cases of contexts. For preliminary experiments we have used the annotated XM2VTS [20] database that provides clean face images and we split it using the Lausanne protocol I in training, validation and testing subsets. On the training subset we applied the MCT-based face detector [1] implemented with Torch3vision and we extracted and plot some of these context features as shown in Fig. 3.

To assign a detection (and its context) to the positive or negative class we have used the Jesorsky measure [21] (see Section VI-B for more details) with a relaxed threshold

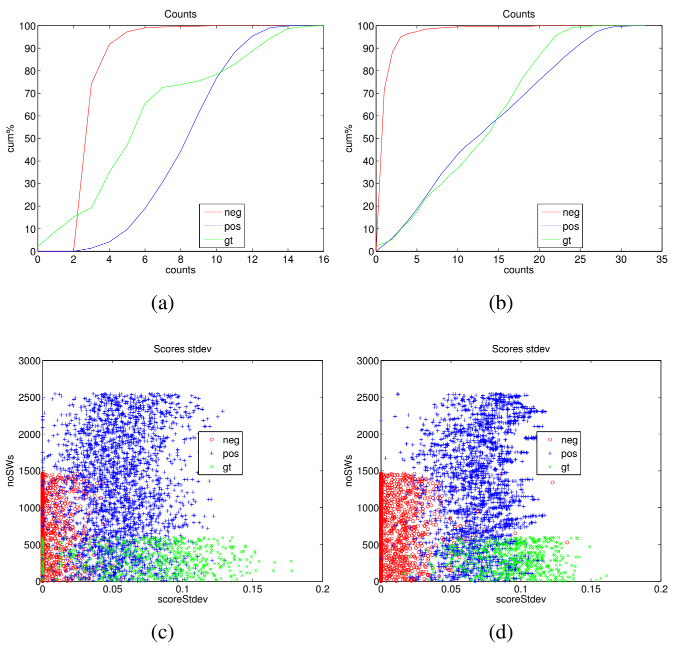


Fig. 3. Distributions of various features using the full versus axis sampling on XM2VTS training dataset (negative - red, positive - blue, ground truth - green). We have used the multiscale approach and the MCT boosting cascade classifier described in [1] and implemented with Torch3vision. Cumulative histogram of counts for two axes (y, scale) using axis sampling (a) and full sampling (b). Cloud points of score standard deviation for 3 axis (x, y, scale) using axis sampling (c) and full sampling (d).

$\epsilon_J = 0.5$. This value is much higher than the reported $\epsilon_J = 0.25$ value used for evaluating face detections, allowing the detections that do not overlap much with the ground truth to be considered as positive detections too. This allows the detections that are outside the Jesorsky distance from the ground truth to be moved to within the Jesorsky distance using the proposed method described in Section V-A. We refer to this as *recovering misaligned detections*.

As it can be seen from Fig. 3, there is a significant difference between the negative and the positive contexts that support our intuition: around a true detection many more detections are generated than around a false alarm. This implies that just by counting detections good discriminative information is obtained. For example in Fig. 3 (a, b) more than 95% of the negative contexts have the count less than 95% of the positive ones. Also, fewer detections implies much less score variation (see Fig. 3 (c, d)) for negative samples. It can be noticed that negative contexts are more compact around the center, while the positive are much more spread having the standard deviation much higher for the combination of two axes (and others).

The full sampling method provides more discriminative information than the axis sampling (see Fig. 3 - (a) versus (b), (c) versus d). This was confirmed by our experiments where it was found that the full sampling context-based model outperformed the axis sampling context-based model.

C. Classifier

The context features computed as described in the previous section are used to train a classifier to distinguish between false alarms and true detections based on their context. We build a linear classifier for each context feature (described in Section IV-C1) and then we combine them to produce the final result (described in Section IV-C2).

Our aim is to *automatically* select the best features and axes that are more discriminant. This makes the context-based model independent of the specific geometric properties of the object to detect, the type of the object classifier or the scanning procedure. This way our method becomes general enough to be applied to a variety of object detection problems.

1) Context classifiers: The contextual information is used to form 35 different context features: there are $n = 5$ types (as discussed in Section IV-B) computed for each of the $m = 7$ axis combinations. For each feature we build a logistic linear model which we denote as $M(x, w)$, where the x is the d-dimensional ($1 \leq d \leq 3$, depending on the attribute type) sample feature vector and w is the $d+1$-dimensional parameter value. The model output is:

$$M(x, w) = \frac{1}{1 + exp\left(-w_0 + \sum_{i=1}^{d} x_i w_i\right)}, \tag{1}$$

where w_0 is sometimes called the bias term and the w_i terms are the weights of the inputs.

Training the model is done by minimizing a specified error function. The error function for logistic models is usually the negative of the likelihood of the model output being generated from the input data. Additional L_1 and L_2 norm regularization terms are added, as described in [22], to make the model more robust. Following [23], our function to optimize is:

$$E(w, \lambda_1, \lambda_2) = \frac{\sum l(w, x^+)}{N^+} + \beta \frac{\sum l(w, x^-)}{N^-} + \tag{2}$$

$$\lambda_1 \underbrace{\sum_{i=1}^{n} |w_i|}_{L_1} + \lambda_2 \underbrace{\sum_{i=1}^{n} |w_i|^2}_{L_2}, \tag{3}$$

where $l(w, x) = -y \, log(M(x, w)) - (1-y) \, log(1 - M(x, w))$ is the negative log likelihood of the sample x using the model weights w; obviously y relates to the label of interest so it represents the positive class for the case of $l(w, x^+)$ and the negative class for $l(w, x^-)$. The log likelihoods are averaged separately over the N^+ positive samples and the N^- negative samples respectively. λ_1 and λ_2 are priors for the L_1 and L_2 norms. The purpose of the L_2 norm regularization term is to avoid over fitting, while the L_1 one is to keep the model sparse hopefully by automatically selecting the most informative features.

The weight β represents the relative importance attributed to the error caused by the negative samples relative to the one caused by the positive samples. In the case of object detection (in particular face detection) it is preferred to have higher false alarms than to miss objects. This implies that β needs to penalize false rejections more than false alarms which corresponds to $\beta < 1$. Several preliminary experiments were performed on a small sub-set of the training data and $\beta = 0.3$ was chosen as the optimal value.

It is rather unusual that the error function $E(w, \lambda_1, \lambda_2)$ is split in two normalized parts for the positive and the negative samples. The motivation for doing this is the unbalanced nature of the training samples. Many more negative samples are generated than the positive samples when the threshold of the face classifier is low and vice versa when the threshold is high. This makes training very hard and the results would be biased towards the category with more samples. By using the *category-based error normalization* as proposed we enforce each category to have the same relative importance no matter how many samples are provided for each one. In our preliminary tests this approach proved much more robust (to biased training data either towards positive or negative samples) than the usual error formulation.

There are some robust methods to optimize the non-continuously differentiable function $E(w, \lambda_1, \lambda_2)$ (for a review see [22]). We have used a simple method called *grafting* described in [23]. This method integrates well with standard convex optimization algorithms and it uses an incremental approach to feature selection that suits our needs. Jorge Nocedal's L-BFGS library [24] was used for the optimization of the error function at each step of the grafting algorithm.

Another related problem we need to solve is the choice of the λ_1 and λ_2 prior terms. For this we use a cross-validation technique on two datasets, one for training and one for tuning, as specified by each database's protocol. We first optimize the λ_1 prior term using a logarithmic scale keeping $\lambda_2 = 0$ and second we optimize the λ_2 prior term using the same logarithmic scale and keeping the already estimated λ_1 value.

The criterion to choose the best (λ_1, λ_2) configuration is the Weighted Error Rate (WER) defined as:

$$WER(\beta, \tau) = \frac{\beta \times FAR + FRR}{\beta + 1}, \tag{4}$$

where FAR is the False Acceptance Rate and FRR is the False Rejection Rate computed as $FRR = 1 - TAR$; TAR is the True Acceptance Rate also referred to as Detection Rate (DR). The same weight β was used as in Equation 3. The WER is optimized by choosing an appropiate threshold $0 < \tau < 1$ of the model (steps 4 and 11 in Algorithm 1). The minimum WER value specifies the best the model can perform and we consider it is a good estimation of the performance of the (λ_1, λ_2) configuration.

We summarize the training in Algorithm 1. In our case the features have low dimensionality (see Table I). This implies that no L_1 norm regularization term is needed and we set $\lambda_1 = 0$ and no λ_1 tuning was performed. However, this term will be used for the combined classifier as shown in the next section.

2) Combined classifier: Each feature classifier can be considered as an expert. By combining them two benefits can be obtained: first the combined classifier should perform better and second only some (the best) experts are combined which implies that some irrelevant features can be (automatically) discarded. The combined model uses the same logistic linear model as for the context classifiers. This makes the proposed hierarchical model a non-linear mapping of the inputs, while each context classifier is kept very simple and linear.

Algorithm 1 Feature/Combined classifier training

```
1: $WER^* = 1, \lambda_1^* = 0, \lambda_2^* = 0, \beta = 0.3$
2: for $\lambda_1 \in \{10^z \| z \in \mathbb{Z}\}$ do
3:    minimize $E(w, \lambda_1, 0)$ using grafting and L-BFGS
4:    minimize $WER(\beta, \tau)$
5:    if $WER(\beta, \tau) < WER^*$ then
6:       $\lambda_1^* = \lambda_1, w^* = w, WER^* = WER(\beta, \tau)$
7:    end if
8: end for
9: for $\lambda_2 \in \{10^z \| z \in \mathbb{Z}\}$ do
10:   minimize $E(w, \lambda_1^*, \lambda_2)$ using grafting and L-BFGS
11:   minimize $WER(\beta, \tau)$
12:   if $WER(\beta, \tau) < WER^*$ then
13:      $\lambda_2^* = \lambda_2, w^* = w, WER^* = WER(\beta, \tau)$
14:   end if
15: end for
16: return weights $w^*$ for the logistic regression
```

The inputs to the combined classifier are the normalized outputs of the context classifiers. Let us define the context classifiers as $M_{k,l}(x, w)$, where k indicates the attribute type ($k = 1..n, n = 5$) and l corresponds to the axis combination ($l = 1..m, m = 7$). Let $\tau_{k,l}$ be the optimum threshold value of the $M_{k,l}$ model. Then the value forwarded to the combined classifier is $x_{k,l} = M_{k,l}(x, w) - \tau_{k,l}$.

This normalization has two benefits. First, the sign indicates the decision of the $M_{k,l}$ model: positive for true detections and negative for false alarms. Second, the absolute value is (empirically) proportional to the confidence of the $M_{k,l}$ model in its decision.

The combined classifier is trained like the small classifiers as shown in Algorithm 1 with the difference that the L_1 norm regularization term is used in this case. This way only the most informative features are automatically selected and combined.

V. Context-based clustering

In this section we propose a new method to cluster multiple detections in two steps. First we refine detections using the same contextual information as described in Section IV and second we cluster multiple detections using the Adaptive Mean Shift algorithm.

A. Refinement of detections

In this section we describe our method for refining the object detections obtained with a generic classifier. We make use of the contextual information as described in Section IV. We argue that the context - the $C(T_{sw})$ collection of 4-dimensional points, provides enough information to assess where the true object location may reside.

Intuitively if some detection slightly overlaps with a true location, but it is translated or covers also some background, its context should contain more detections on the side that covers better the true location. The current detection should therefore be moved more to that specific side, which is given by clustering the context positions where detections occur. This method is presented in Algorithm 2.

Algorithm 2 Context-based detection refining

```
Require: context-based model $CM$
Require: $D = T_{sw}$ - the list of detections
 1: $R = \emptyset$
 2: for $T_{sw} \in D$ do
 3:     $T_{sw}^0 = T_{sw}, t = 0, valid = true$
 4:     while $\|T_{sw}^{t+1}, T_{sw}^t\| < \epsilon, t < T, valid$ do
 5:         build $C(T_{sw}^t)$
 6:         apply $CM$ on $C(T_{sw}^t)$
 7:         if $T_{sw}^t$ is not a false alarm then
 8:             $T_{sw}^{t+1} = Predict(C(T_{sw}^t))$
 9:             $t = t + 1$
10:         else
11:             $valid = false$
12:         end if
13:     end while
14:     if valid then
15:         $R = R \cup T_{sw}^t$
16:     end if
17: end for
18: return $R$
```

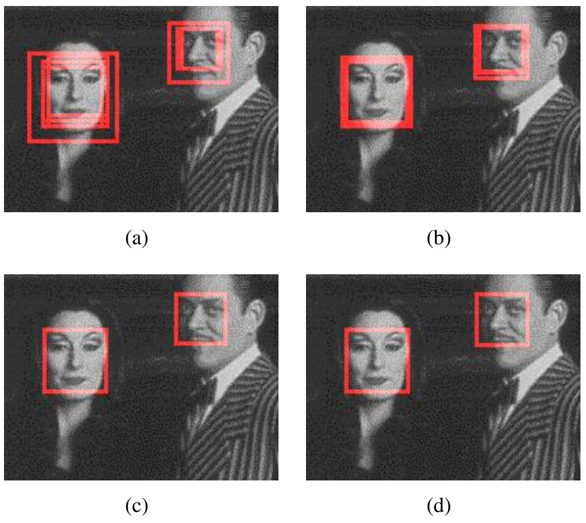


Fig. 4. An example of context-based detection refining using an image in the MIT+CMU dataset. We have used the multiscale approach, the MCT boosting cascade classifier described in [1] and the full context sampling. (a) after running the context-based model. (b), (c), (d) - after 1, 2 and 10 iterations.

The algorithm is initialized with a context-based model CM and a set of detections D, for instance built using the multiscale sliding window approach and the same object classifier used to train CM. The output is a new set of detections R that should not contain false alarms but only improved object locations.

Each detection T_{sw} from D is verified against CM. If it is validated, the $Predict$ method applied on its context provides a new location where to iteratively move T_{sw}. In our experiments we have used a very simple $Predict$ function as the average of the (x_i, y_i, s_i) components of $C(T_{sw})$. The algorithm is stopped when: i) T_{sw} converges (no significant displacement in the sub-window location and scale between two consecutive iterations) or ii) the maximum number of iterations is reached or iii) it is invalidated by CM.

Fig. 4 presents some results of our method on an image in the MIT+CMU dataset. If no refinement is performed (T_{sw} remains fixed), but only the false alarms are removed using CM, we obtain the results in Fig. 4 (a). After refining the detections for several iterations (Fig. 4 (b, c, d)) the detections get closer and closer to the true face location and they converge to almost the same position. Please note that the true detections are all kept and they are only moved to a better position, although they may seem pruned in this example.

Preliminary experiments have shown that the convergence is very fast, in most cases just a couple of iterations. We have set the maximum number of iterations as $T = 10$ in all our experiments.

B. Clustering multiple detections

The previous section has shown that the detection refinement method makes the problem of clustering multiple detections much easier (see Fig. 4 - d vs. a). This is very important because the typical merging algorithms (e.g. Adaptive Mean Shift, averaging, non-maxima suppression) processes just a list of detections and their scores. There is no good metric to state precisely what should be merged and what should be removed using only this information. This is why some heuristics are usually used such as: removing isolated detections or smaller sub-window than an application specific constant or merging together the ones overlapping more than a threshold. But our context-based model and the detection refinement method do not need these heuristics and they work well with any merging algorithm.

Still we have chosen the (Adaptive) Mean Shift algorithm for several reasons. First it is non-parametric and it does not require the number of clusters (the number of objects) to be known or properly initialized. Second the adaptive version works well for clusters of significantly different sizes (in our case objects of different sizes that appear in the same image). The last reason is that from our knowledge this is the first attempt to apply this method for clustering detections.

Here we shortly describe the Adaptive Mean Shift (AMS) algorithm and the modifications required to process object detections. It is a non-parametric estimator of density gradient with many applications such as (high-dimensional) data clustering, edge preserving filtering, image segmentation and tracking. For more details, consult [25]–[27].

Assuming a collection of data points $x_i \in R^d, i = 1, ..., n$, each one having an associated bandwidth h_i (giving the *adaptive* property), the sample point estimator of the density (of some unknown density function f) at point $x \in R^d$ is given by:

$$\hat{f}_k(x) = \frac{1}{n} \sum_{i=1}^{n} \frac{1}{h_i^d} k \left(\left\| \frac{x - x_i}{h_i} \right\|^2 \right), \tag{5}$$

where k is a non-negative kernel function radially symmetric centred in zero and integrating to one. Let us define the function $g(x) = -k'(x)$ where the derivative of the kernel exists.

Following [27], the mean shift vector for an Euclidean space is defined as:

$$m_G(x) = \frac{\sum_{i=1}^{n} \frac{x_i}{h_i^{d+2}} g\left(\left\|\frac{x-x_i}{h_i}\right\|^2\right)}{\sum_{i=1}^{n} \frac{1}{h_i^{d+2}} g\left(\left\|\frac{x-x_i}{h_i}\right\|^2\right)} - x. \tag{6}$$

which gives the direction of maximum increase in the density. This is the direction of the basin of attraction (cluster center), where the density gradient becomes zero. Applying (6) iteratively for each point separately will make each one converge to its cluster in a finite number of steps.

The bandwidths represent the area of influence of each point. They must be estimated prior to the actual clustering, usually by investigating each data point's neighbourhood using some distance metric. It is shown in [27] that the variable bandwidth approach gives better results than the fixed bandwidth variant.

The specific problem of AMS clustering of multiple detections requires estimating the bandwidth for each detection. We represent the detections as 4D points like (x_i, y_i, w_i, h_i) - the top-left coordinates and the width and height of the sub-window. Then the bandwidth is set as the distance between a target sub-window and the farther sub-window that still overlaps with it. If no sub-window is overlapping with the target than its bandwidth is set to 0. This prevents the isolated detections to be falsely removed and we rely on the context-based classifier to prune such detections.

VI. Experimental procedure

The experimental procedure used in this paper is defined by these aspects: the image databases, the associated protocol, the face detection procedure and the method for evaluating the detection performance.

A. Databases

We performed extensive experiments using multiple databases: XM2VTS [20], BANCA [28], WEB [12], CINEMA [12] and MIT+CMU [3]. The XM2VTS and BANCA databases contain one large centered face in each image taken in a controlled environment and are commonly used in face verification experiments. The last three databases are considered the most difficult because they consist of images with multiple, sometimes very small, degraded faces or without any face, taken in different environments (indoor and outdoor).

The XM2VTS database is split using the Lausanne Protocol I in training, validation and testing datasets, while the others are used only for testing (see Table II). Each image is manually annotated with the eye centers which is used for evaluating the detection results.

Type	**Database**	**#Images**
Train	XM2VTS	600
Validation	XM2VTS	800
Test	XM2VTS	960
Test	BANCA (English)	6240
Test	WEB	211
Test	CINEMA	158
Test	MIT + CMU	117

TABLE II
Number of images for each dataset.

B. Face detection

Our method was tested using a popular off-the-shelf face classifier. The MCT-based boosted cascade face classifier [1] implemented with the Torch3vision open-source library [18] is used without any modifications.

We alter the performance of the face classifier by varying the threshold (θ) of the last stage of the cascade. This allows us to understand if the performance of the classifier affects the performance of the context models. For each θ two context-based models have been trained: using both *full* and *axis* context sampling methods.

Varying the threshold θ can lead to a severe miss-match in the number of positive and negative samples. Setting the threshold too high would output no face detections (or very few) which gives no data to train the context-based model with, while setting it too low makes the training dataset biased towards the negative samples. We have chosen empirically the threshold interval as to obtain at least hundreds of both positive and negative training samples.

The detections (and contexts) are obtained using a standard sliding-window approach. It should be noted that no sub-window heuristic pruning was used during scanning. Indeed, too smooth or too noisy sub-windows are typically rejected before being tested with the face classifier so as to improve the speed and sometimes decrease the number of false alarms.

In all our experiments we have used the multiscale sliding-window approach. The scanning parameters used are: the scale sampling coefficient $s_s > 1$ that defines the relative sub-window size increase between two scales and the position sampling coefficients $0 < s_x, s_y < 1$ relative to the current sub-window size.

Three different configurations were experimented with: fine ($s_s = 1.1$ and $s_x = s_y = 0.05$), medium ($s_s = 1.1$ and $s_x = s_y = 0.1$) and coarse ($s_s = 1.2$ and $s_x = s_y = 0.2$). The purpose of using different scanning configurations is to test the robustness of our method. Note that most applications usually require a medium or coarse scanning configuration in order to achieve real-time performance. The scanning parameters influence the number of sub-windows to evaluate (see Table III) and implicitly the speed of the detection.

Resolution	**Parameters**	**No. sub-windows**
fine	$s_s = 1.1, s_x = s_y = 0.05$	3091392
medium	$s_s = 1.1, s_x = s_y = 0.1$	685251
coarse	$s_s = 1.2, s_x = s_y = 0.2$	86475

TABLE III
Number of sub-windows to evaluate for each scanning resolution on a 800x600 image.

We have compared our method with the different multiple detection merging and false alarms pruning methods implemented with Torch3vision: Non-Maxima Suppression (NMS) as the merging step of the algorithm described in Section II-C and Adaptive Mean Shift (AMS) as presented in Section V-B. We also report results and normalize the evaluation measures to the case of using no merging at all (NoMerge).

To label a detection as positive we used the Jesorsky measure (see next paragraph) with the threshold $\epsilon_J = 0.25$ [21]. Note that this is more restrictive than the usual condition to register a positive detection: a minimum overlap of 50% with the ground truth bounding box [2]. This implies that the reported TAR values may be smaller than the latest reported state-of-the-art results. But we are motivated into using the Jesorsky measure because it provides a better alternative to compare the accuracy of the detections obtained with different methods (see Section VII-C).

Following [21] this measure defines how much both eye detections vary from the ground truth relative to the distance between the eyes. Let us define the detected eye positions as l_d and r_d (left and right) and the ground truth positions as l_g and r_g. The errors in pixels of each detection are $E_L = \|l_d, l_g\|$ and $E_R = \|r_d, r_g\|$ and the distance between the eyes D.

Then the Jesorsky error is defined as: $\epsilon_J = \frac{max(E_L, E_R)}{D}$, which linearly relates the maximum allowed eye distance error with the distance between the eyes.

C. Detection performance evaluation

To present the face detection results we analysed the TAR and the number of false alarms (FA). These performance measures are parametrized by the threshold of the face classifier: $TAR = TAR(\theta)$ and $FA = FA(\theta)$ respectively. We omit the threshold of the context-based classifier in this parametrization because it is automatically optimized on the tuning dataset (see Section IV-C) and it is not varied during experiments.

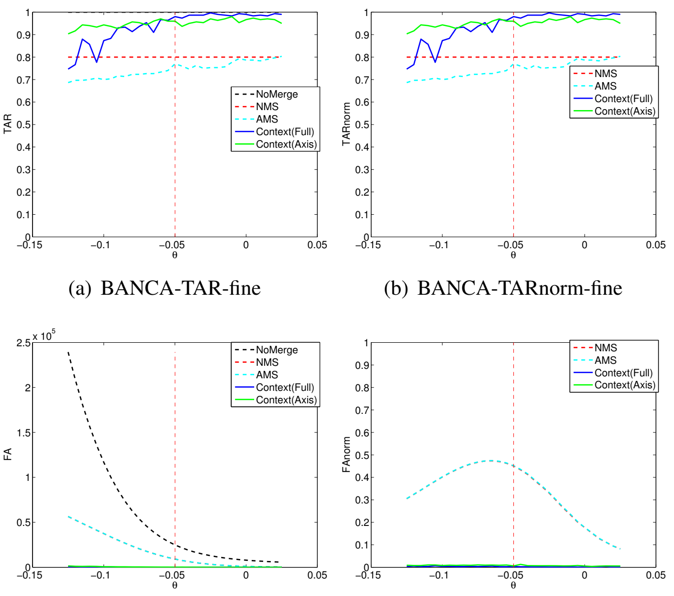
The typical method of comparing different models is to use a Receiver Operating Characteristic (ROC) curve that describes the relation between the TAR and the FAR (or the FA) by varying its *discriminative threshold*. The higher the area under the curve is, the better a model performs which corresponds to a high TAR with a low FAR (or FA) [29]. In our case the threshold to vary is θ, which is not the discriminative threshold. Indeed it just provides different context distributions to train our context-based model.

A more appropriate evaluation method is the Expected Performance Curve (EPC) [29]. The EPC provides unbiased estimates of performance at various operating points and it makes possible to compare our proposed method with a baseline. It also eliminates the restriction that θ is the discriminative threshold as required for the ROC curve.

Let us consider two measures $V_1(\theta)$ and $V_2(\theta)$ parametrized by the same parameter θ, for example the FAR and the FRR. In our case θ is the threshold of the last stage of the face classifier. These two measures are usually linearly combined in a single function that will make the comparison easier:

$$C(\alpha) = \alpha V_1(\theta) + (1-\alpha) V_2(\theta). \tag{7}$$

$\alpha \in \{0, 1\}$ is the trade-off coefficient that is task specific. It has a similar meaning as the β coefficient use in Eq. 3 and Eq. 4 and the following relation can be expressed: $\alpha = \frac{\beta}{\beta+1}$. A small α favours the models with higher TAR and with arbitrary FAR, while a large α favours the models with lower FAR and with arbitrary TAR. In between, the models with a good trade-off between TAR and FAR are considered better.



(a) BANCA-TAR-fine (b) BANCA-TARnorm-fine

(c) BANCA-FA-fine (d) BANCA-FAnorm-fine

Fig. 5. The effect of normalizing the TAR (top row) and the FA (bottom row) on the BANCA dataset. NMS and AMS have very similar FA values, but with significantly different TAR values. The default threshold point is represented with dashed red vertical line.

The EPC curve is built using the function C and varying the trade-off coefficient α. For each α value the model parameter that minimizes the function C is computed as:

$$\theta = \underset{\theta}{\operatorname{argmin}}\, C\left(V_1(\theta), V_2(\theta), \alpha\right), \tag{8}$$

on a tuning dataset. Then on a different test dataset the C is plotted against α, with the same θ computed on the tuning dataset.

For the face detection case the V_1 and V_2 are naturally chosen as the FAR and the FRR. This may be misleading because we compare in our paper multiple detections clustering methods (context-based and baseline) and not face classifiers. The aim of any multiple detection clustering algorithm is to remove as few as possible true detections and remove as many as possible false alarms.

This implies that the TAR obtained *after* clustering should be linearly normalized to the TAR of the face classifier without any merging. But we cannot proceed similarly with the FAR because the denominator (the number of detections) is changing for each θ. We use instead the logarithmically normalized FA. This is because the FA obtained with our method is much smaller than of the baseline method (Fig. 5 c) while the TAR has similar values (Fig. 5 a).

Let us define the TAR and the FA of the face classifier without any merging (NoMerge as in Fig. 5) as TAR_n and FA_n respectively. Then the function C used for comparing the models is defined as:

$$C(\alpha) = \frac{\alpha}{\log(2)} \log\left(1 + \frac{FA(\theta)}{FA_n(\theta)}\right) + \tag{9}$$

$$(1-\alpha)\left(1 - \frac{TAR(\theta)}{TAR_n(\theta)}\right). \tag{10}$$

VII. Results

This section presents the results obtained using our proposed method on several popular face databases as described in Section VI. We have evaluated the following aspects:

- the context-based model (see Section IV) - how well it distinguishes false alarms from true detections,
- the face detection performance using the proposed context-based system and
- the accuracy of the detections.

A. *Context-based model evaluation*

In the first set of the experiments we have evaluated how well the context-based model distinguishes between false alarms and true detections. For this we have computed and plotted the TAR and the WER, as shown in Fig. 6, for all scenarios using the fine scanning settings. Each row of plots corresponds to one testing datasets (XM2VTS, BANCA, MIT+CMU), with the plots on the left presenting the TAR and the plots on the right presenting the WER evolution. The full and axis sampling situations are plotted on the same graphic to easily compare them.

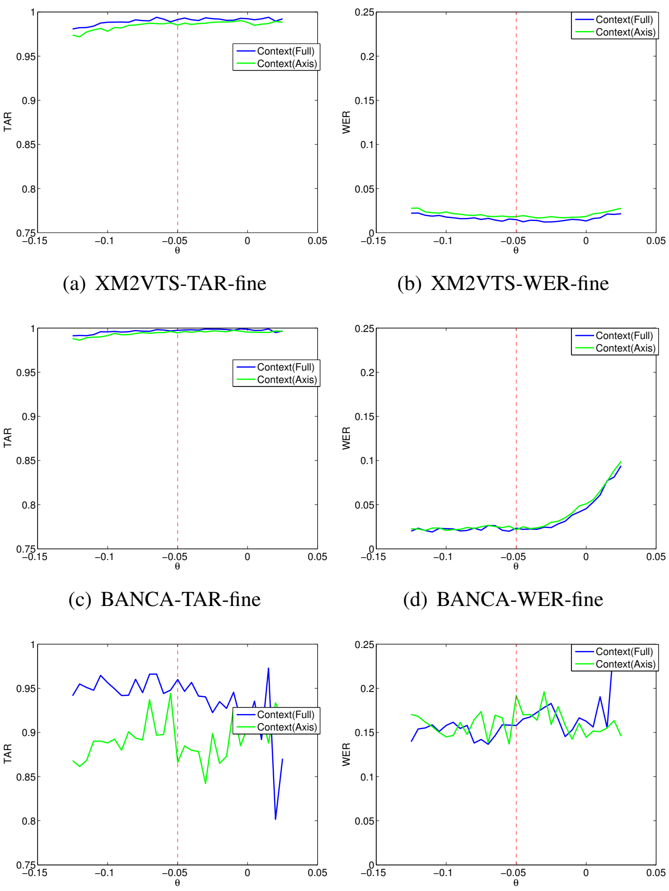
The full sampling context-based model performs better than the axis sampling one for the majority of different threshold values. However, there it requires much bigger contexts (see Section IV-A) which decreases the speed of the overall face detection process. Even with the axis context sampling our context-based model manages to distinguish well enough the false alarms from true detections.

Both full and axis sampling models have a TAR higher than 95% with a WER smaller than 5% for the XM2VTS and BANCA scenarios - see Fig. 6 (a, b, c, d). The performance decreases for the more challenging MIT+CMU scenario - see Fig. 6 (e, f), but with at most 10% relative to XM2VTS and BANCA. It is important to note that the training data becomes very scarce for high θ and in these situations the context-based model is expected to perform worse (higher WER).

These results are stable across multiple threshold values of the face classifier and indicates that the features extracted from the contexts are discriminative enough, although very simple and low dimensional (see Section IV-B). We conclude that the contextual information provides rich enough information to accurately classify false alarms. In the next sections we analyse how our model improves the face detection results.

B. *Face detection evaluation*

The face detection system represented in Fig. 2 was evaluated in the next set of experiments. In these experiments we studied the effect of: i) using either the Non-Maxima Suppression (NMS) or the Adaptive Mean Shift (AMS) baseline



(a) XM2VTS-TAR-fine (b) XM2VTS-WER-fine

(c) BANCA-TAR-fine (d) BANCA-WER-fine

(e) MIT+CMU-TAR-fine (f) MIT+CMU-WER-fine

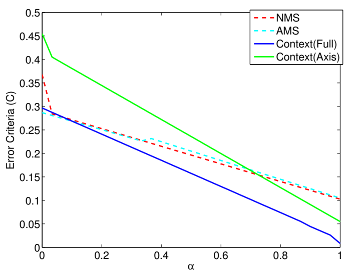
Fig. 6. Context-based model results (TAR and WER) on XM2VTS (a, b), BANCA (c, d) and MIT+CMU (e, f) datasets using fine scanning. The default threshold point of the face classifier is represented with dashed red color.

methods and ii) using the context-based system with (CtxFull) full or axis (CtxAxis) sampling. To compare these methods we have used the unbiased EPC as discussed in Section VI-C, plotted as shown in Fig. 7.

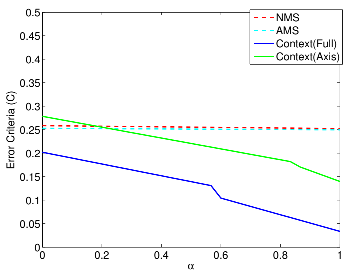
As shown in Fig. 7 there is a significant performance variation between the three methods across all the dataset and with different scanning precisions. The full sampling context-based model (CtxFull) consistently outperforms the others, while the axis sampling variant (CtxAxis) performs on average better than the baseline methods only for medium and coarse scanning.

The coarse and medium scanning settings produces a significant number of misaligned detections (treated as false alarms, because they are outside ϵ_J from the ground truth) which cannot be recovered by the baseline methods. Instead our context-based model refines these detections to become true detections. This results in a significant decrease in the number of false alarms, which implies that the error criteria C (See Eq. 10) decreases accordingly for large values of the trade-off parameter α (see Fig. 7 c, f, i): from $C > 0.5$ for the baseline methods to $C < 0.10 - 0.15$ for the context-based methods.

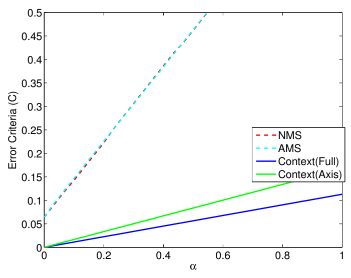
Some detection examples on the MIT+CMU database using fine scanning are presented in Fig. 8. Our method successfully removes the false alarms, while keeping all the true detections.



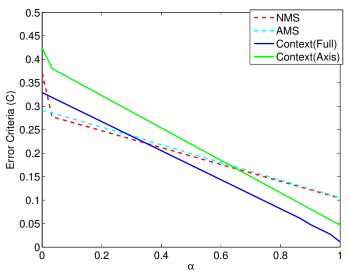
(a) WEB-fine



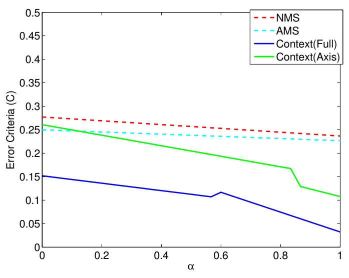
(b) WEB-medium



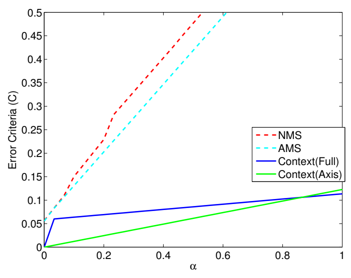
(c) WEB-coarse



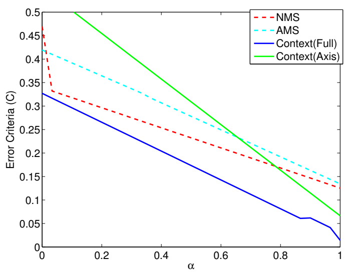
(d) CINEMA-fine



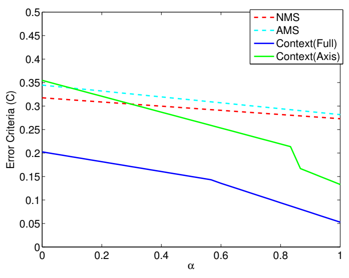
(e) CINEMA-medium



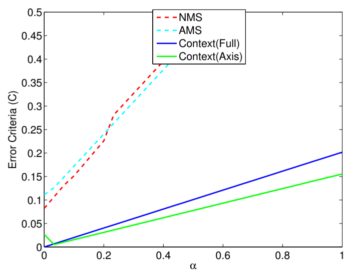
(f) CINEMA-coarse



(g) MIT+CMU-fine



(h) MIT+CMU-medium



(i) MIT+CMU-coarse

Fig. 7. Unbiased EPC plots on WEB (a, b, c), CINEMA (d, e, f) and MIT+CMU (g, h, i) datasets using fine (left column) and medium (right column) scanning. Low α value favourites models with higher TAR, while higher α - models with lower FAR. Mid-range α value are considered a good trade-off between TAR and FA.

To better understand the effect of changing the scanning resolution, we have also plotted the TAR variance for various scanning settings (see Fig. 9). We argue that our method successfully recovers misaligned detections by moving a wrong detection (outside ϵ_J from the ground truth) to a better location (within ϵ_J). We dynamically sample around this detection to assess, independently of other detections, if it is actually a false alarm. In most situations the context-based model will accept it and the detection refining algorithm will move it closer to the ground truth (within ϵ_J). This way in some cases we report TAR even higher than using no merging at all, while the baseline methods always decrease the TAR (see Fig. 9).

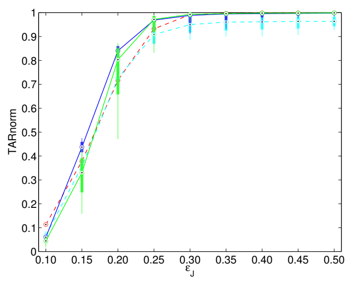
Another important aspect is that our method has a TAR that decreases much slower with the scanning resolution than of the baseline methods. For example the CtxFull method presents a TAR decrease of approximately 10% at the default threshold for the BANCA dataset from the fine to coarse resolution, while the NMS and AMS methods decrease with 40% (see Fig. 9 b vs. f). An important speed-up can therefore be achieved: it is possible to significantly decrease the scanning resolution, thus processing fewer sub-windows, while maintaining the same level of performance using the baseline methods but with a finer scanning resolution.

On average our proposed method can be up to three times slower than using NMS or AMS. This is because for each detection, a fixed number of sub-windows have to be evaluated. Still in most cases these sub-windows can be evaluated multiple times (mainly because of detection refinement algorithm). Our implementation does not use this fact, but it can be improved by caching the already evaluated sub-windows. The time difference can be further reduced using a lower scanning resolution as previously discussed.

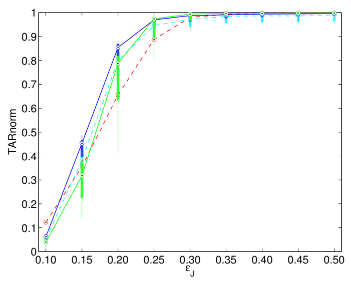
C. *Accuracy evaluation*

Next we have studied the precision of the face detections to assess the impact of the clustering method presented in Section V. For this we have plotted the normalized TAR for various threshold values of the Jesorsky measure: small values correspond to precise detections while large values to imprecise detections. We have used increments of $\delta\epsilon_J = 0.05$ from $\epsilon_J = 0.10$ to $\epsilon_J = 0.50$.

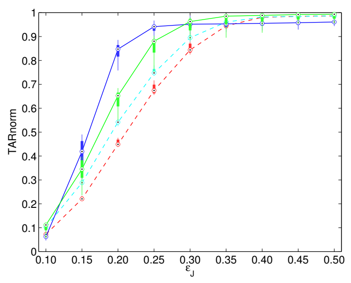
The normalized TAR variation with the threshold of the Jesorsky measure is represented in Fig. 10. The lines pass through the median of the normalized TAR for all θ values at



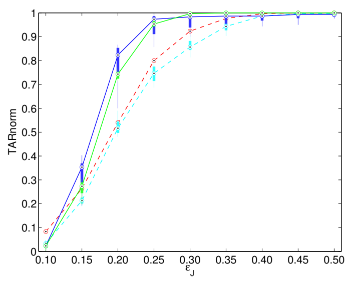
(a) XM2VTS-fine



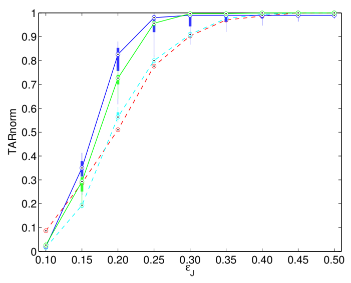
(b) XM2VTS-medium



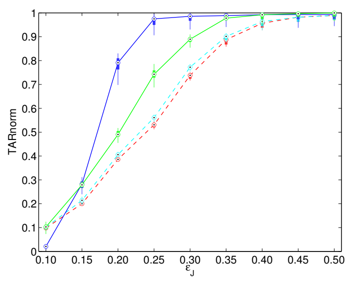
(c) XM2VTS-coarse



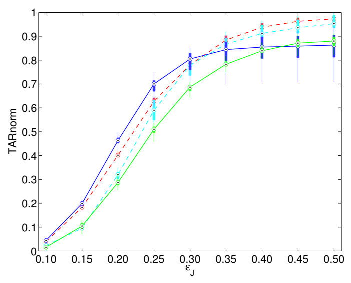
(d) BANCA-fine



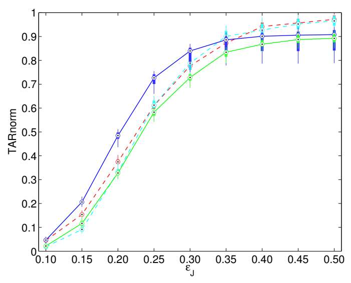
(e) BANCA-medium



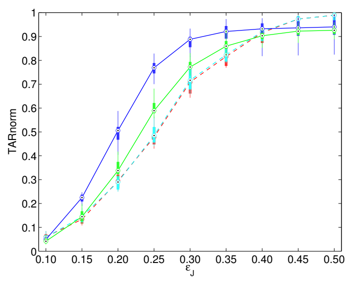
(f) BANCA-coarse



(g) MIT+CMU-fine



(h) MIT+CMU-medium



(i) MIT+CMU-coarse

Fig. 10. Normalized TAR for various Jesorsky thresholds on WEB (a, b, c), CINEMA (d, e, f) and MIT+CMU (g, h, i) datasets using fine, medium and coarse scanning. The results using the NMS merging method are represented with dashed red, using the AMS with dashed magenta, while using the context-based model with blue (full sampling) and with green (axis sampling). The horizontal axis represents the Jesorsky threshold, while the vertical axis the normalized TAR.

a specific ϵ_J value, with the range of +/-25% from the median being represented with a filled box.

The full sampling method (Fig. 10 - blue) consistently outperforms the baseline (Fig. 10 - red) for all scenarios - it's TAR saturates faster. For example at the relative low Jesorsky measure threshold of $\epsilon_J = 0.20$, our method detects 10%, 50% and 30% more detections on the MIT+CMU, XM2VTS and BANCA scenarios respectively using the fine scanning. This indicates that our context-based method generates significantly more accurate detections. This is useful for face localization and face verification algorithms as they need to be initialized with as precise detections as possible.

The axis sampling method (Fig. 10 - green) is less accurate than the full sampling version. Still it has a significantly higher TAR than the baseline for both the XM2VTS and BANCA datasets. On the more challenging MIT+CMU dataset its performance slightly degrades with up to 5%.

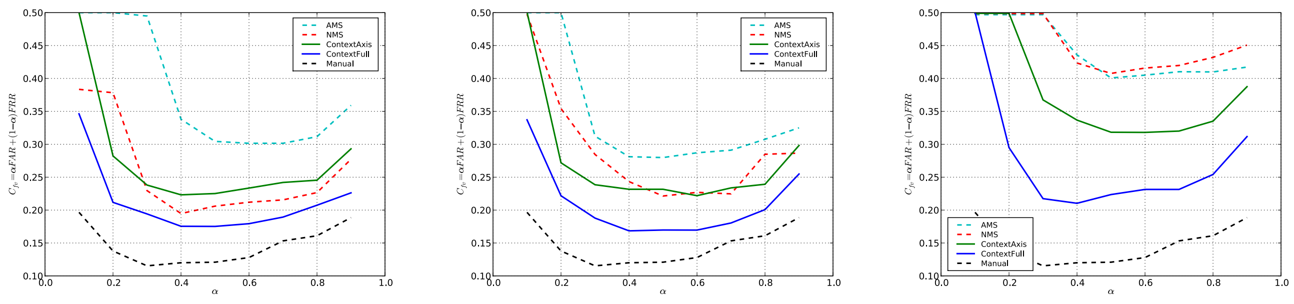
We have also performed some face verification experiments to further assess the improvement in detection accuracy. For this, we have used a well-known face verification algorithm [30] implemented using Torch3vision. The BANCA database was split using the Protocol P [28] and the images with no detected face were excluded, as it is commonly done in automatic face verification experiments, independently for each method.

The unbiased EPC curves are presented in Fig. 11 for AMS, NMS, the two context-based models and the manual annotation case (Manual). It can be noticed that the detections obtained with our context-based method are significantly more useful for face verification. The improvement is much more prominent for the medium and coarse scanning, even for the axis sampling variant. This result validates once more the conclusion that our method is less sensitive to the scanning resolution obtained in the previous section.

VIII. Conclusion

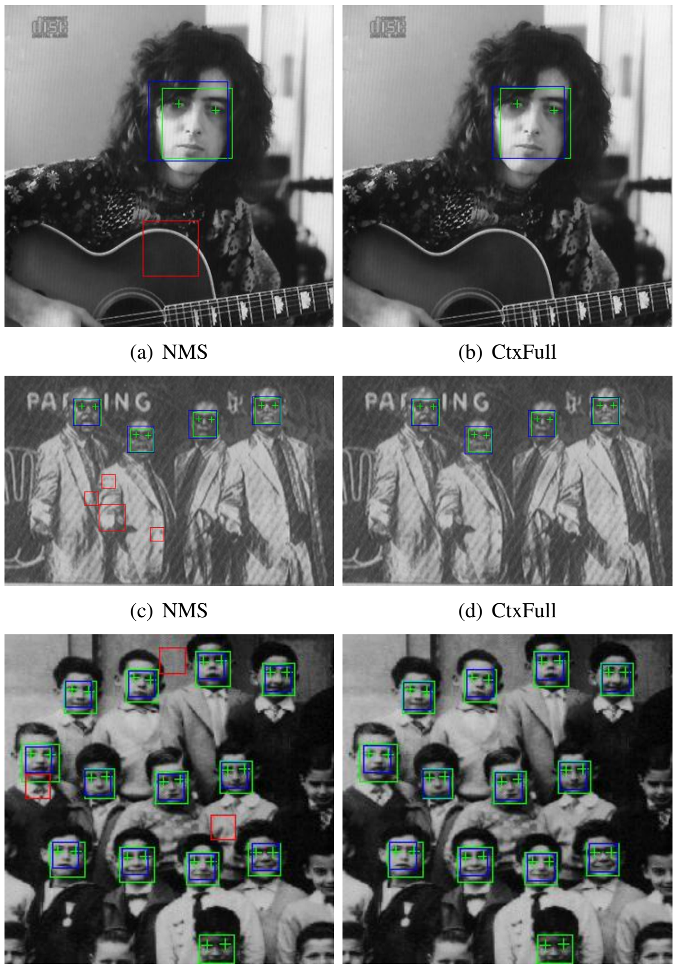
This paper has presented a new post-processing technique for object detection that models the output (context) of an object classifier. We focused on processing multiple detections and false alarms in a more principled way and less sensitive to the scanning resolution than the baseline methods.

Experiments on several popular face databases, using a popular face classifier, have shown a significant increase in



(a) BANCA-fine (b) BANCA-medium (c) BANCA-coarse

Fig. 11. Unbiased face verification EPC plots on BANCA dataset using fine, medium and coarse scanning. The results for manual annotation are represented with dashed black line.

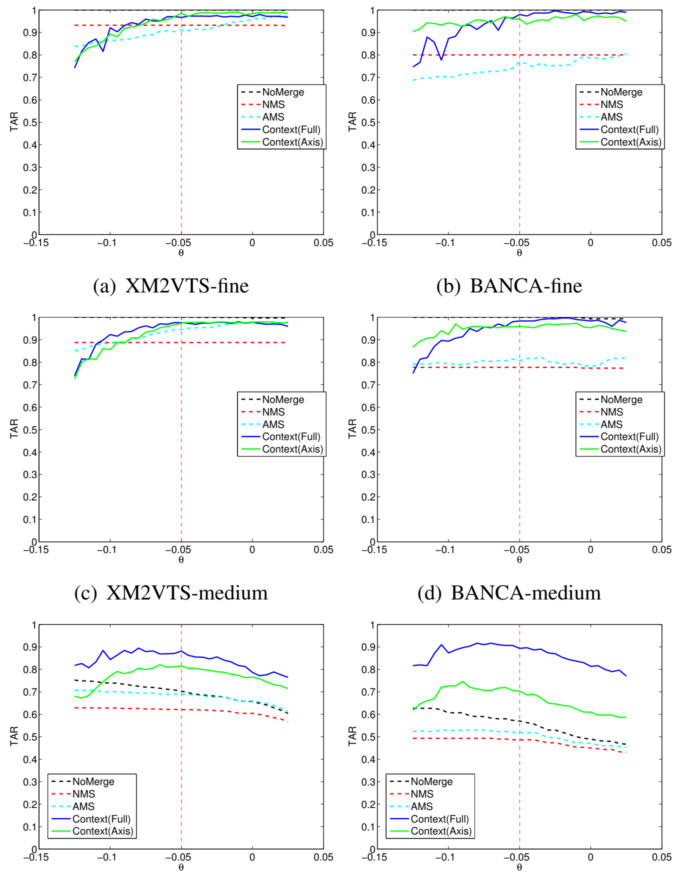


(a) NMS (b) CtxFull
(c) NMS (d) CtxFull
(e) NMS (f) CtxFull

Fig. 8. Detections on the MIT+CMU database using fine scanning: in the left column using the NMS baseline method and in the right column using the context-based model with full sampling. The ground truth bounding boxes are represented with green, the true detections with blue and false alarms with red.

performance (higher TAR, exponential lower FA, detections' accuracy).

Our approach makes several contributions to the object (face) classifier field. First we have shown that it is possible to build a simple non-parametric model that learns to distinguish between false alarms and true detections with high accuracy. Second we have proposed an iterative method that uses the contextual information to generate more accurate detections. Finally we have used the Adaptive Mean Shift to cluster multiple detections.



(a) XM2VTS-fine (b) BANCA-fine
(c) XM2VTS-medium (d) BANCA-medium
(e) XM2VTS-coarse (f) BANCA-coarse

Fig. 9. TAR on XM2VTS (a, c, e) and BANCA (b, d, f) scenarios using fine (top row), medium (middle row) and coarse (bottom row) scanning.

There are several additional advantages of using our method. First it is more robust to the scanning accuracy - its performance degrades much slower with the scanning resolution than the baseline methods. Second it makes the task for the clustering algorithm much easier because multiple related detections are merged using our detection refinement method to some very close positions and false alarms are removed successfully. Third the more accurate detections we obtained improve the performance of the face localization and verification processes and allows us to recover some imprecise detections. Finally our experiments have shown that the context (checked successfully against the context-based model) gives the right direction where to shift such detections within the Jesorsky threshold. Another important advantage is that our algorithm can be initialized with any sub-window collection and it does not make any assumption on the face classifier's score values, its type or the features used.

Still further improvements can be made by: designing more powerful possible higher dimensional context-based features, using more powerful non-linear feature classifiers and designing a more efficient sampling method.

Acknowledgment

The authors would like to thank the FP7 European MOBIO project (IST-214324) and the Hasler Foundation (CONTEXT project) for their financial support and to Laurent El Shafey (Idiap) for his help with the face verification experiments.

References

[1] B. Froba and A. Ernst, "Face detection with the modified census transform," *IEEE International Conference on Automatic Face and Gesture Recognition*, pp. 91–96, 2004.

[2] P. Viola and M. Jones, "Rapid object detection using a boosted cascade of simple features," *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 1, pp. 511–518, 2001.

[3] H. A. Rowley, S. Baluja, and T. Kanade, "Neural network-based face detection," *IEEE Transactions On Pattern Analysis and Machine intelligence*, vol. 20, pp. 23–38, 1998.

[4] C. H. Lampert, M. B. Blaschko, and T. Hofmann, "Beyond sliding windows: Object localization by efficient subwindow search," in *Computer Vision and Pattern Recognition (CVPR)*, 2008, pp. 1–8.

[5] M. B. Blaschko and C. H. Lampert, "Learning to localize objects with structured output regression," in *European Conference on Computer Vision (ECCV)*, vol. 5302, 2008, pp. 2–15.

[6] H. Takatsuka, M. Tanaka, and M. Okutomi, "Spatial merging for face detection," in *SICE-ICASE International Joint Conference*, 2006, pp. 5587–5592.

[7] ——, "Distribution-based face detection using calibrated boosted cascade classifier," in *ICIAP '07: Proceedings of the 14th International Conference on Image Analysis and Processing*, 2007, pp. 351–356.

[8] Y. Freund and R. Schapire, "A short introduction to boosting," *Journal of Japanese Society for Artificial Intelligence*, vol. 14, no. 5, pp. 771–780, 1999.

[9] Y. Freund, "An adaptive version of the boost by majority algorithm," *Machine Learning*, vol. 43, no. 3, pp. 293–318, 2001.

[10] R. E. Schapire, "The boosting approach to machine learning: An overview," in *Nonlinear Estimation and Classification*, D. D. Denison, M. H. Hansen, C. Holmes, B. Mallick, and B. Yu, Eds. Springer, 2003.

[11] R. Feraud, O. Bernier, J.-E. Viallet, and M. Collobert, "A fast and accurate face detector based on neural networks," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 23, no. 1, pp. 42–53, 2001.

[12] C. Garcia and M. Delakis, "Convolutional face finder: A neural architecture for fast and robust face detection," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 26, no. 11, pp. 1408–1423, 2004.

[13] R. Lienhart, A. Kuranov, and V. Pisarevsky, "Empirical analysis of detection cascades of boosted classifiers for rapid object detection," in *25th Pattern Recognition Symposium*, Madgeburg, Germany, 2003, pp. 297–304.

[14] T. Ojala, M. Pietikainen, and D. Harwood, "A comparative study of texture measures with classification based on feature distributions," *Pattern Recognition*, vol. 29, no. 1, pp. 51–59, January 1996.

[15] S. Z. Li and Z. Zhang, "Floatboost learning and statistical face detection," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 26, no. 9, pp. 1112–1123, 2004.

[16] J. Wu, J. Rehg, and M. Mullin, "Learning a rare event detection cascade by direct feature selection," in *Advances in Neural Information Processing Systems 16*. MIT Press, 2003.

[17] C. Huang, H. Ai, B. Wu, and S. Lao, "Boosting nested cascade detector for multi-view face detection," in *17th International Conference on Pattern Recognition*, vol. 2. Washington, DC, USA: IEEE Computer Society, 2004, pp. 415–418.

[18] Y. Rodriguez, "Face detection and verification using local binary patterns," Ph.D. dissertation, Ecole Polytechnique Fdrale de Lausanne, 2006.

[19] B. Froba, A. Ernst, and C. Kublbeck, "Real time face detection," in *Signal and image processing: Proceedings of the fourth IASTED international conference*, 2002.

[20] K. Messer, J. Matas, J. Kittler, J. Lttin, and G. Maitre, "Xm2vtsdb: The extended m2vts database," in *In Second International Conference on Audio and Video-based Biometric Person Authentication*, 1999, pp. 72–77.

[21] O. Jesorsky, K. J. Kirchberg, and R. Frischholz, "Robust face detection using the hausdorff distance," in *AVBPA '01: Proceedings of the Third International Conference on Audio- and Video-Based Biometric Person Authentication*. London, UK: Springer-Verlag, 2001, pp. 90–95.

[22] S. in Lee, H. Lee, P. Abbeel, and A. Y. Ng, "Efficient L_1 regularized logistic regression," in *Association for the Advancement of Artificial Intelligence (AAAI)*, 2006.

[23] S. Perkins, K. Lacker, J. Theiler, I. Guyon, and A. Elisseeff, "Grafting: Fast, incremental feature selection by gradient descent in function space," *Journal of Machine Learning Research*, vol. 3, pp. 1333–1356, 2003.

[24] D. C. Liu and J. Nocedal, "On the limited memory bfgs method for large scale optimization," *Mathematical Programming*, vol. 45, pp. 503–528, 1989.

[25] D. Comaniciu and P. Meer, "Mean shift analysis and applications," in *The Proceedings of the Seventh IEEE International Conference on Computer Vision, 1999.*, 1999, pp. 1197–1204.

[26] B. Georgescu, I. Shimshoni, and P. Meer, "Mean shift based clustering in high dimensions: a texture classification example," in *Proceedings of the Ninth IEEE International Conference on Computer Vision, 2003.*, 2003, pp. 456–463.

[27] D. Comaniciu, "An algorithm for data-driven bandwidth selection," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 25, pp. 281–288, 2003.

[28] E. Bailly-Bailliére, S. Bengio, F. Bimbot, M. Hamouz, J. Kittler, J. Mariéthoz, J. Matas, K. Messer, V. Popovici, F. Porée, B. Ruiz, and J.-P. Thiran, "The banca database and evaluation protocol," 2003, p. 1057.

[29] S. Bengio, J. Mariéthoz, and M. Keller, "The expected performance curve," in *In Proceedings of the 22nd International Conference on Machine Learning*, 2005, pp. 9–16.

[30] W. Zhang, S. Shan, W. Gao, X. Chen, and H. Zhang, "Local gabor binary pattern histogram sequence (lgbphs): A novel non-statistical model for face representation and recognition," *Computer Vision, IEEE International Conference on*, vol. 1, pp. 786–791, 2005.