

A sparsity-enforcing method for learning face features

Augusto Destrero ¹, Christine De Mol ², Francesca Odone ^{1*}, Alessandro Verri ¹

¹ DISI, Università di Genova,

Via Dodecaneso 35 I-16146 Genova, Italy

{destrero, odone, verri}@disi.unige.it

² Université Libre de Bruxelles

boulevard du Triomphe, 1050 Bruxelles, Belgium

demol@ulb.ac.be

** corresponding author*

Abstract

In this paper we propose a new trainable system for selecting face features from over-complete dictionaries of image measurements. The starting point is an iterative thresholding algorithm which provides sparse solutions to linear systems of equations. Although the proposed methodology is quite general and could be applied to various image classification tasks, we focus here on the case study of face and eyes detection. For our initial representation, we adopt rectangular features in order to allow straightforward comparisons with existing techniques. For computational efficiency and memory saving requirements, instead of implementing the full optimization scheme on tenths of thousands of features, we propose a three-stage architecture which consists in finding first intermediate solutions to smaller size optimization problems, then merging the obtained results, and next applying further selection procedures. The devised system requires the solution of a number of independent problems, and hence the necessary computations could be implemented in parallel. Experimental results obtained on both benchmark and newly acquired face and eyes images indicate that our method is a serious competitor to other feature selection schemes recently popularized in computer vision for dealing with problems of real-time object detection. A major advantage of the proposed system is that it performs well even with relatively small training sets.

I. INTRODUCTION

The last decade has seen an increasing use of over-complete, general-purpose sets of features for representing the visual information contained in images [7], [17], [23]. Loosely inspired by biological systems [19] such sets, coupled with appropriate learning techniques, appear to provide an effective alternative to higher-level descriptions for object detection and recognition [21], [33]. While in higher-level approaches the prior knowledge is embedded in the feature description, in trainable methods based on general-purpose features the prior knowledge is confined instead in the preparation of the training set. Since only a relatively small fraction of the over-complete set of features – typically a set of huge cardinality – is relevant for a given problem, the latter methods have to address the issue of feature selection.

Feature selection is a problem of formidable complexity. While a coherent theory is still missing (see [11] for a review of the main approaches to feature selection in machine learning), a number of trainable methods are emerging in the empirical practice due to their effectiveness [29], [35], [38]. Among them it is worth mentioning the impressive amount of work on the use of Adaboost methods [9] for feature selection [10], [16], [33], [37]. An alternative and interesting way to cope with feature selection in a learning-by-examples framework is to resort to sparsity-enforcing regularization techniques involving a L1-norm constraint or penalty term [3], [29]; this route is widely unexplored for learning problems dealing with images.

In this paper (which was presented in preliminary form at the ACCV07 conference [5]), we propose to use for feature selection in computer vision a penalized version of the so-called Lasso regression [29]. This choice is driven by three major considerations: first, a simple iterative algorithm for computing the corresponding solution has been recently proposed in [4]. Second, this feature selection mechanism has not been to date applied and evaluated in the context of vision, where typically many spatially- and highly-correlated features are involved. Finally, we want to confront the mathematical framework analyzed in [4] with empirical practice and assess its merits and limitations, hoping that the gained experience will stimulate further theoretical and numerical progress in the field. This paper makes a step in this direction by proposing an extension of this framework able to cope with the high dimensionality and other specificities of feature selection for computer vision.

In order to allow for a direct comparison of our approach with state-of-the-art methods, we have decided

to focus our research on a case study, namely face detection. To this aim, we use so-called rectangle features [33] (widely retained as a very good starting point for many computer vision applications) as a starting representation and support vector machines [31] as classifiers (although results obtained with Adaboost are also included).

Our results demonstrate that the sparsity-enforcing approach to feature selection we propose is very effective for computer vision applications. As a side effect we also gain a deeper understanding of the properties of the adopted feature-selection framework *per se*. Our approach uses a resampling strategy coupled with a multi-stage feature selection to cope with very large numbers of features.

The final set of selected features not only leads to the construction of state-of-the-art classifiers but also has been further optimized to build a hierarchical real-time system for face detection operating in a busy corridor and using a very small number of loosely correlated features. Experimental results on the problem of detecting eyes in images, simply obtained by replacing the *face* training set with an *eye* training set, show that the devised approach can also be easily extended to other visual detection tasks.

We now briefly discuss related work in the field. In the last years face detection has been boosted by the contribution of the learning-from-examples paradigm which helps solve many well-known problems related to face variability in images [36]. Early approaches to view-based face and object detection were based on the whole image [20], [22], [25], [26], [28]. Later on, component-based approaches highlighted the fact that local areas are often more meaningful and more appropriate to cope with occlusions and deformations [13], [18]. There is now little doubt that, in order to obtain effective and efficient face (and object) detection, it is advisable to identify a small set of meaningful features for each specified object class; image representations based on pixels are simply inadequate to take into account structure, symmetries, and co-occurrences in the class of interest. Many types of features have been proposed and are used for faces: let us mention rectangle features [33], wavelet representations [21], Ranklets [8], and many others. All these representations produce huge descriptions which may include a very large number of irrelevant or redundant features. In order to make the use of such representations practical, it is advisable to identify subsets of meaningful features. To this purpose an automatic feature selection procedure is desirable.

Hierarchical methods have been widely studied as speed-up strategies for object detection. They are based on the quite general assumption that most background elements are very different from the class

of interest, and also that usually the background covers most of the image area. Literature on hierarchical approaches to face detection includes [2], [6], [12], [24], [33]. Coarse-to-fine procedures are related to – and take inspiration from – focus-of-attention approaches [14], [30], according to which it is often possible to determine rapidly the position of objects in a scene, leaving more elaborate processing for the most promising regions.

Our method may seem very closely related to the work [33], which will constitute our basic reference point; nevertheless a closer analysis reveals that the similarities between the two methods are very few: we start off with the same initial representation, but while in [33] the feature selection procedure is intertwined with the classification stage, in our approach feature selection and classification (performed with SVMs) are sequential tasks. In addition, the experimental analysis we carried out shows that our method achieves comparable results with a much smaller training set — and in particular with a very limited number of negative examples. In this respect our method is perhaps more closely related to the paper [12], even if we automatically compute the layers of the cascade while [12] relies on a selection of classifiers designed ahead, implementing decision surfaces of increasing complexity. Finally, our approach has connections with the approach in [34], which combines the use of rectangle features and SVMs but does not consider either automatic feature selection or cascading.

The paper is organized as follows. In Section II we describe the sparsity-enforcing algorithm which will be the main ingredient of our feature selection method. In Section III we introduce the application scenario. The two-stage feature selection method is illustrated and discussed in Section IV whereas the face detector system built on top of the feature selection process is described in Section V. Section VI contains our conclusions.

II. FEATURE SELECTION THROUGH ITERATIVE SOFT-THRESHOLDING

In this section, we describe the basic algorithm on which our feature selection method is built upon. We restrict ourselves to the case of a *linear* dependence between input and output data, which means that the problem can be reformulated as the solution of the following linear system of equations:

$$\mathbf{g} = \mathbf{A}\mathbf{f} \quad (1)$$

where $\mathbf{g} = (g_1, \dots, g_n)^\top$ is the $n \times 1$ vector containing output labels, $\mathbf{A} = \{A_{ij}\}, i = 1, \dots, n; j = 1, \dots, p$ is the $n \times p$ matrix containing the collected features j for each image i and $\mathbf{f} = (f_1, \dots, f_p)^\top$

the vector of the unknown weights to be estimated.

Since in the present context the dimensions of \mathbf{A} are large, even if we had $n = p$, the usual approaches for solving the algebraic system (1) turn out to be unfeasible. Moreover, typically the number of features p is much larger than the dimension n of the training set, so that the system is hugely under-determined. Because of the redundancy of the feature set, we also have to deal with the collinearities responsible for severe ill-conditioning. Both difficulties call for some form of regularization and can be obviated by turning the problem (1) into a penalized least-squares problem. Classical regularization such as the so-called *ridge regression* (also referred to as Tikhonov's regularization) uses a quadratic penalty, typically the L^2 -norm of the vector \mathbf{f} : $\|\mathbf{f}\|_2^2 = \sum_j |f_j|^2$. Such quadratic penalties, however, do not provide feature selection in the sense that the solution of the resulting penalized least-squares problem will typically yield a vector \mathbf{f} with all weights f_j different from zero. This is the reason why the replacement of quadratic penalties by sparsity-enforcing penalties has been advocated in recent literature; such penalty should enforce automatically the presence of (many) zero weights in the vector \mathbf{f} . The L^1 -norm of \mathbf{f} , which is the sum of the absolute values of the weights, i.e. $|\mathbf{f}|_1 = \sum_j |f_j|$, is a suitable penalty in order to enforce sparsity while preserving convexity of the optimization problem. Hence we solve, instead of (1), the following penalized least-squares problem:

$$\mathbf{f}_L = \arg \min_{\mathbf{f}} \{ \|\mathbf{g} - \mathbf{A}\mathbf{f}\|_2^2 + 2\tau |\mathbf{f}|_1 \} \quad (2)$$

where τ is a positive parameter regulating the balance between the data misfit and the penalty (the so-called *regularization parameter*). In variable selection problems, this parameter also allows to vary the degree of sparsity (number of true zero weights) of the vector \mathbf{f} . After [29], problem (2) is usually referred to as “Lasso regression” (an acronym for Least Absolute Shrinkage and Selection Operator). Whereas ridge regression solutions depend linearly on the output vector \mathbf{g} , the L^1 -norm penalty makes the dependence of the lasso solutions on \mathbf{g} nonlinear. Hence the computation of L^1 -norm penalized solutions is more difficult than with L^2 -norm penalties. In the special case where $\mathbf{A}^\top \mathbf{A}$ is the identity matrix the solution of (2) is easily seen to be $\mathbf{f}_L = \mathbf{S}_\tau(\mathbf{A}^\top \mathbf{g})$ where \mathbf{S}_τ is the following “soft-thresholder”, widely used in wavelet-based denoising schemes, and acting componentwise on a vector \mathbf{h} :

$$(\mathbf{S}_\tau \mathbf{h})_j = S_\tau h_j \quad \text{where} \quad S_\tau h_j = \begin{cases} h_j - \tau \operatorname{sign}(h_j) & \text{if } |h_j| \geq \tau \\ 0 & \text{otherwise} \end{cases} \quad (3)$$

Hence the parameter τ appears as a threshold value, under which a component is set to zero. The number of selected features (non-zero weights) is controlled by the threshold τ . When $A^\top A$ is different from the identity, there is no longer a closed-form expression for the minimizer (2) and several numerical strategies have been proposed in the literature to solve the corresponding and rather cumbersome nonlinear optimization problem. We adopt in this paper a simple iterative strategy, namely the following scheme

$$\mathbf{f}_L^{(t+1)} = \mathbf{S}_\tau[\mathbf{f}_L^{(t)} + A^\top(\mathbf{g} - \mathbf{A}\mathbf{f}_L^{(t)})] \quad t = 0, 1, \dots \quad (4)$$

with an arbitrary initial vector $\mathbf{f}_L^{(0)}$. In the absence of soft-thresholding (or equivalently for $\tau = 0$) this scheme is sometimes referred to as the Landweber iteration and converges to the generalized solution (minimum-norm least-squares solution) of (1). The iterative soft-thresholding algorithm scheme (4) has been proved in [4] to converge to a minimizer of (2), provided the norm of the matrix \mathbf{A} is renormalized to a value strictly smaller than 1. Notice that since the L^1 -norm is not strictly convex, the uniqueness of such minimizer cannot be guaranteed in the presence of a nontrivial null-space of \mathbf{A} (and in particular when the number of features is larger than the number of examples).

III. SETTING THE SCENE

In this Section we briefly introduce the application scenario we consider: a monitoring system installed in a busy corridor of our department. Then we describe the features at the basis of our initial representation and explain why a feature selection procedure is needed.

A. The application scenario

The application motivating our work is a face detector to be integrated in a monitoring system installed in our department. Therefore most of our experiments are carried out on a dataset of images collected by that system. The system monitors a busy corridor acquiring video shots when motion is detected. We used the acquired video frames to extract examples of faces and non-faces (non-faces are motion areas containing everything but a face). We crop and rescale all images to the size 19×19 — as done in the publicly available CBCL-CMU database of faces and non-faces frontal image patches, which we also use in some of the experiments.

Our scenario produces few difficult negative examples, while face examples are quite challenging since faces rarely appear in a frontal position. In addition, the quality of the signal is low due to the fact that the

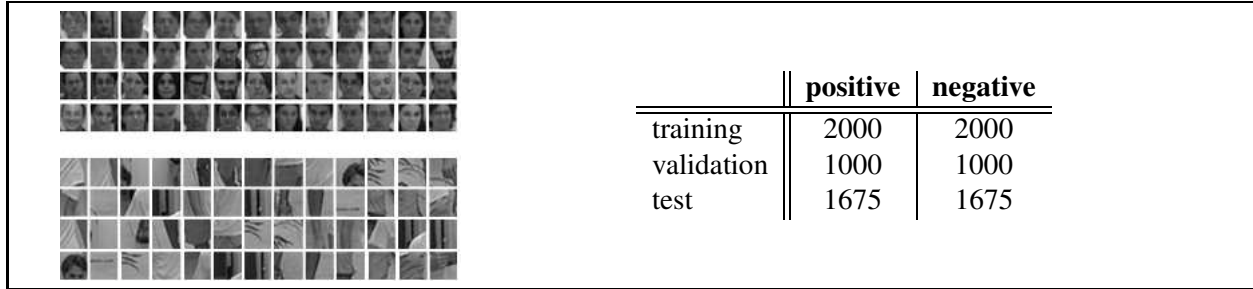


Fig. 1. Examples of positive (top) and negative (bottom) data gathered with our monitoring system. The table reports the size of the sets we used for our experiments.

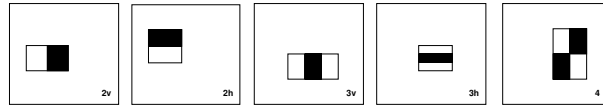


Fig. 2. The support of the 5 types of *rectangle features*. The value of each image feature is then obtained by subtracting the gray levels of the pixels falling in the white areas from the gray levels of the pixels falling in the dark areas.

acquisition device is a common video-surveillance camera and the detected object are often affected by motion blur. Figure 1 shows examples of positive and negative training images and specifies the size of the dataset collected and used for our experiments. We also have at our disposal a bigger set of negative examples (currently comprising about 50 000 image patches), which we used to test our resampling strategies in the case of large datasets.

B. The initial representation

Our training data are examples of faces and of non-faces. Raw data are image patches of size $N \times N$. Instead of using the image patches directly, we represent their content by means of *rectangle features*, as described in [33]. Empirical analysis showed that for face detection rectangle features, although more primitive, compare favorably to other local filters, such as steerable filters [33]. Let us remark that other features capturing similar structures, such as Haar basis functions (used in [21] for object detection) and Ranklets [27] (used in [8] for face detection), could also be used as starting representations for the methodology proposed in the present paper. Our choice of the rectangle features relies on the following considerations: (a) they encode the typical structure of faces, which are made of various kinds of symmetries and peculiar brightness patterns, (b) they can be implemented efficiently through an

intermediate image representation called the “integral image”, and (c) they allow us to easily compare the performances of our feature selection scheme with those of alternative methods using the same initial set of features. For more details on such features we refer the interested reader to [33].

Figure 2 shows the support of the 5 different types of rectangle features we are using: the value of each image feature is obtained by subtracting the grey-levels of pixels falling in the white areas from the gray-levels of pixels falling in the dark areas. We call the five groups of features $\mathcal{F}_{2h}, \mathcal{F}_{2v}, \mathcal{F}_{3h}, \mathcal{F}_{3v}, \mathcal{F}_4$ — the subscripts referring to the feature type.

We compute such features over different locations, sizes, and aspect ratios for each $N \times N$ image patch, obtaining in such a way an over-complete set of image descriptors. When taking $N = 19$ (as in our experiments with faces), we compute about 64000 rectangle features per patch. Given the size of the resulting image description obtained, if we had to compute the whole set of rectangle features for each analyzed image patch, the task would be impossible in the case a video analysis: then, clearly, some kind of dimensionality reduction has to be performed. In view of such applications, the goal of our study is, therefore, to find a well-defined procedure to extract a small group of relevant features that can be computed at run time for each analyzed image patch.

IV. THE PROPOSED FEATURE SELECTION ALGORITHM

In the present Section we discuss in full detail the method we propose for the feature selection task. Our starting point is the iterative soft-thresholding algorithm described in Section II. However, due the high dimensionality of the problem, we have to modify the basic scheme to preserve computational efficiency and cope with memory requirements. Instead of implementing the full optimization scheme using the entire set of features, we first solve a number of smaller size optimization problems obtained by randomly subsampling the feature vector. Then we compile a list of selected features by picking up the features that have been selected each time they appear in a subproblem. We provide strong empirical evidence showing that the solution obtained in this way is indeed close to the solution of the full optimization scheme. We then show that a more effective list of selected features (a few percent of the original features) can be obtained by applying again the same feature selection algorithm to the list obtained in the first stage. The results we obtain are carefully analyzed. Finally, we present a third selection step which further reduces the features to a small number of almost independent features, now perfectly suitable for real-time detection. The price to pay for this third shrinking of the list of selected features is a slight decrease

in performance, but such decrease appears to be negligible for most practical purposes.

A. Sampled version of the iterative soft-thresholding scheme

We first specialize the algorithm described in Section II to our case study, then we discuss implementation issues and report the results of our experimental analysis. Finally, we comment on the need for an additional feature selection stage to be introduced in Section IV-C.

1) *Algorithm specialization:* We start by considering the problem in the form (1) where \mathbf{A} is the n by p matrix of the processed image data, the entry A_{ij} representing the j^{th} rectangle feature obtained from the image labelled by i . The data matrix is manipulated in order to center the features values around their median and then normalized by dividing its entries by a number slightly larger than its largest eigenvalue (this is done in order to guarantee convergence of the iterative scheme, see Section II). Since we are working in a binary classification setting we associate to each datum (image) a label $g_i \in \{-1, 1\}$. Each entry of the unknown vector \mathbf{f} is associated to one feature. Performing feature selection is equivalent to looking for a sparse solution $\mathbf{f} = (f_1, \dots, f_p)^\top$ of (1), i.e. for a weight vector that has many zero entries. This is done through the use of the iterative algorithm (4). Features corresponding to non-zero weights f_i are retained as relevant to discriminate between the two classes. After checking empirically that the choice of the initialization vector was not a crucial issue, we chose to always initialize the weight vector \mathbf{f} with zeroes: $\mathbf{f}^{(0)} = \mathbf{0}^\top$. Once the matrix is correctly prepared (see herebelow), the iterative scheme is very simple to implement, as demonstrated by Table I which provides a pseudo-code describing the matrix construction, centering and normalization, and the iterative scheme (4). We use the following stopping rule for the iteration according to the stability of the solution reached: at the t -th iteration we evaluate $\|\mathbf{f}^{(t)} - \mathbf{f}^{(t-1)}\|_2$: if this quantity is smaller than some prescribed threshold T (which we choose to be a fraction of the value of the norm $\|\mathbf{f}^{(t)}\|_2$, $T = \|\mathbf{f}^{(t)}\|_2/100$) during 100 consecutive iterations, we conclude that the obtained solution is stable and stop the iterative process.

2) *Implementation and design issues:* We now describe in detail how we build the linear system: each of the 4000 images of our training set is represented by 64000 measurements of the rectangle features. Since these measurements are real-valued and the dimension of the matrix \mathbf{A} is 4000×64000 , the matrix size in bytes is about 1 Gb (if each entry is represented in single precision). For this reason applying the iterative algorithm as given by (4) directly to the whole matrix may not be feasible on all PCs: the matrix multiplication needs to be implemented carefully so that we do not keep in primary memory the

Initialization and data normalization

```
% compute data matrix A
for each image I_i, i=1,...,n
    compute p features F_ij, j=1,...,p
    A(i,j) = F_ij
    y(i)={-1,1} %label associated to I_i
% center data matrix A --- see text
for each column of A, A(i,:)
    A(i,:)=(A(i,:) - median(A(i,:)) / IQR (A(i,:)); %IQR=inter-quartile range
% normalize A --- see text
lambda_max=highest_eigenval(A) %for instance with the power method
for each i,j
    A(i,j)=A(i,j)/lambda_max;
    g(j) = y(j)/lambda_max;
f_i = 0;
```

Thresholded Landweber

```
while (exit conditions are not met) %see text
for i=1:n %see Eq (4)
    f = f + A'(g - A*f);
    %thresholding foreach element of f
    if abs(f(i)) >= tau
        f(i) = f(i) - tau*sign(f(i));
    else f(i) = 0;
    end
end
```

TABLE I

MATLAB-LIKE PSEUDO-CODE OF THE FEATURE SELECTION ITERATIVE PROCEDURE. THE FIRST PART DESCRIBES THE PREPARATION OF THE LINEAR SYSTEM, INCLUDED MATRIX CENTERING AND NORMALIZATION. THE SECOND PART DESCRIBES THE ITERATIVE SCHEME. SEE TEXT.

entire matrix \mathbf{A} . A possible solution to this problem consists in computing intermediate solutions with multiple accesses to secondary memory.

We investigated and implemented a different approach, which goes as follows and is based on a resampling of the features set aiming at splitting the original problem into many problems of smaller size. It consists in building S feature subsets by extracting each time m features from the *original* set of size p ($m \ll p$), thus obtaining S smaller linear subproblems of the type:

$$\mathbf{A}_s \mathbf{f}_s = \mathbf{g} \quad s = 1, \dots, S,$$

where \mathbf{A}_s is a submatrix of \mathbf{A} containing the columns relative to the features in s ; \mathbf{f}_s is defined accordingly.

We need to choose the number S of subproblems and their size. Each time we extract with replacement

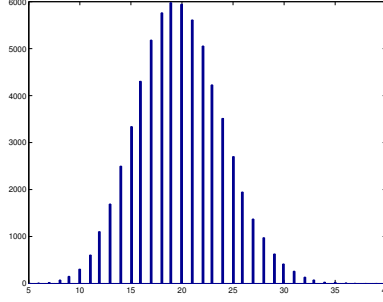


Fig. 3. A histogram of the features showing how frequently they are extracted. The x -axis represents the number of extractions. (see text).

a subset of the original features. The subset size should be big enough to be descriptive, but small enough to produce an easy-to-handle matrix. We consider subsets of 6400 features (10% of the original size). As for the choice of the number of subproblems S , we rely on the binomial distribution and estimate how many extractions are needed in order that each feature is extracted at least 10 times with high probability. We observe that, if $S = 200$, such a probability is 99.6% which is good enough for our purpose. In practice this means that only 256 features over 64000 are extracted less than 10 times. Figure 3 shows an unnormalized histogram of the features indicating how frequently they are extracted. The x -axis represents the number of extractions. The histogram is peaked around 20, and 99.5% of the features have been extracted between 10 and 30 times.

After having build the S subproblems, we look for the S solutions by running the iterative algorithm (4) for each subproblem. At the end of the process we are left with S overlapping sets of features. The final set is obtained choosing the features that have been selected $\alpha\%$ of the times they appear in the subsets. Usually, unless stated otherwise, we choose $\alpha = 100$, namely we only retain the features that have been selected *each* time they appeared in the subset.

One drawback of the resampling strategy is that, when recombining the S solutions at the end, we loose the features ordering provided by the weights. Observing that the relative ordering of the S solutions was consistent (i.e., features had similar ranks in the different sets including them — see Figure 4), we assigned to each feature a weight that is the average of all the weights associated to it in the different solutions. The final features may then be ordered according to these average weights.

The experimental analysis we report on in the next Section confirms that our resampling strategy leads

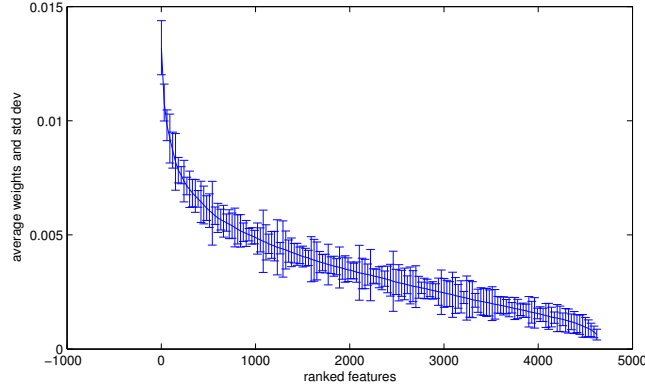


Fig. 4. The selected features S_1 ranked according to their average weight (see text). The figure shows their average weight whereas the bars indicate the individual variations of the weights in the different subproblems.

to results comparable to the original version of the algorithm, but with a substantial gain in efficiency.

As for the choice of the threshold parameter τ (the *model selection* problem) we have considered two alternative methods:

- **Method 1:** choose τ so as to enforce a prescribed number of zeroes in the solution (or, equivalently, to select a prescribed number of features), say z , in about I iterations.
- **Method 2:** choose τ on the basis of the generalization performance, using cross-validation. In particular such choice should provided a good compromise between a small classification error on the validation set and a small number of features. Therefore, we select a certain range of values of τ leading to classification rates below a certain threshold and, among them, choose the value of τ that provides the smallest number of features.

In both cases, since the resampling strategy leaves us with S problems to solve, we tune the parameter on the first subproblem and use the same τ for all subproblems, instead of performing S different choices. We will comment further on this point in the next subsection.

3) *Experimental analysis:* We now report on the experiments we performed using the feature selection process described above. We consider 200 subproblems built each time by extracting 10% of the total set of features.

To evaluate the goodness of the selected features in terms of their generalization ability, we computed the classification performance on a test set of previously unseen data. To this purpose we adopted a

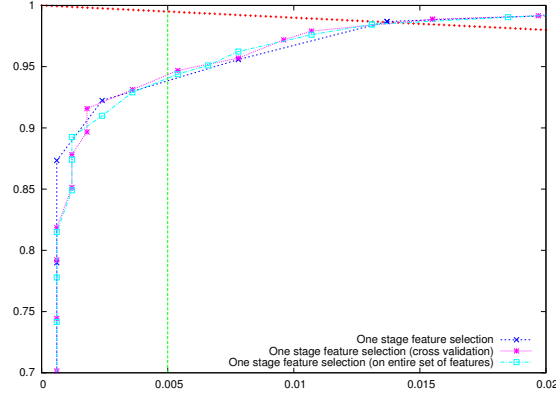


Fig. 5. Comparison between the direct solution of the full problem and the resampling strategy (with two different parameter choices): we observe the generalization performance of a linear SVM on data represented according to set S_1 . The intersection of the ROC with the vertical line gives the hit rate relative to 0.5% false positives, the intersection with $f(x) = 1 - x$ gives the e.e.r.

linear SVM classifier [32] and proceeded as follows. We represented the dataset images according to the selected features and defined a training set, a validation set and a test set. Then we trained the SVM on the training data, tuned the SVM regularization parameter on the validation set and, finally, evaluated the classification performance on the test set. To this purpose, we built a ROC curve over the test set, varying the SVM offset b . In particular we analyzed two points of the ROC curve: the classical *equal error rate* (e.e.r.), corresponding to an equal percentage of false positives and false negatives (in our case, an equal number of missed faces and wrongly labelled non-faces), and the error rate corresponding to a *small false positive rate*, useful for the particular case of object detection in which the ratio between positive and negative examples on a random image is small.

Let us first analyze the effectiveness of our resampling strategy by comparing its results with those obtained when solving directly the full problem (1) on a high-performance PC. Figure 5 shows that there is no loss in performance when applying the resampling strategy. We argue that this may be due to the fact that the high correlation among features typical for images allow us to obtain intermediate solutions that are strongly related to the global solution. This result leads us to conclude that the resampling strategy is indeed an effective alternative which makes possible to solve problems involving a very large number of features and examples.

We now compare our two different parameter choice strategies. We first apply Method 1 with a fixed percentage of zeroes: we use the first subproblem to choose a τ leading to 90% of zeros in about 1000



Fig. 6. The first 100 features of the selected group: redundancy is apparent (see text).

iterations. Since with Method 1 the number of zeroes is fixed, the size of the selected feature sets is quite stable: on average 638.7 ± 0.6 over all the S subproblems. Instead, when applying Method 2, we obtained feature sets of average size equal to 194.9 ± 23.4 corresponding to $97 \pm 1\%$ zeroes. Again, we use only the first subproblem for choosing τ by means of cross validation on the validation set.

The decision to tune the threshold parameter on just one subproblem is motivated by computational reasons. Implementing S model selection procedures would be unpractical. Hence we assumed that the subproblems were similar enough for that purpose. While in the case of Method 1 empirical evidence corroborates this assumption (with the value of τ selected on the first problem by setting the number of zeroes, we get very similar number of zeroes for the other subproblems), there is more instability in the case of Method 2. In both cases, after having solved the 200 subproblems, we merge the feature sets obtained, keeping only the features that were selected *each* time they appeared in a given subproblem.

The final set of selected features, S_1 , contained 4636 features for Method 1 and 2368 for Method 2. Hence, in both cases less than 10% of the original feature set was retained. The selected features are a good synthesis of the original description, as confirmed by the classification results on our test set (see Figure 5). Also, they maintain all the required descriptiveness, representing all meaningful areas of a face (Figure 6 shows the support of the first 100 features over 4636, drawn on top of a sample face).

4) *Comments:* Although the reported classification results are satisfactory, the number of selected features is still high — higher than the size of the feature set obtained with Adaboost on our same dataset (about 70 features), higher than the number of principal components (about 400), and also higher than the intrinsic size of original input data (19×19 pixels).

Indeed, in order to assess the intrinsic size of our data, we have applied a principal component analysis

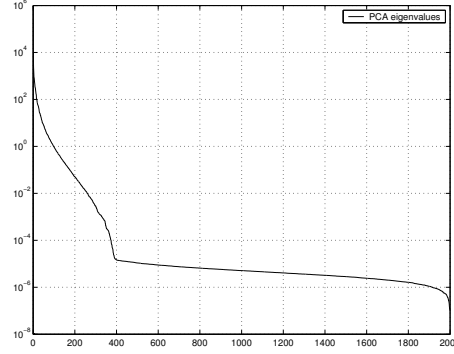


Fig. 7. PCA eigenvalues spectrum, computed on the training images represented by the features of set S_1 . The *elbow* appears before 400 giving us a hint on the intrinsic size of the data.

(PCA) to the set of features selected previously. Even if this dimensionality reduction method is not applicable to our case (since it does not reduce the amount of computations at run time), it can give us hints on the intrinsic size of the analyzed data. Figure 7 shows the eigenvalues spectrum obtained on the training dataset represented with the features of set S_1 . The *elbow* appears before the 400-th eigenvalue, suggesting that even if the image content has been expanded in a redundant high-dimensional set of features, the intrinsic size of the data is close to the number of entries in the original images.

A natural way to further reduce the number of features within the previous framework is to force a higher number of zeros through Method 1. When doing so, however, we noticed that the features descriptiveness decreases considerably. By requiring 99% of zeros in about 1000 iterations, we selected 345 features which by visual inspection showed both a higher degree of short-range correlation and less variety in the selected patterns with respect to the previous outputs. Accordingly, the classification results on the test set dropped of about 3%.

To check whether this decrease in performance is inherent in the choice of the L1 penalty term or is due to the resampling strategy, we also solved Problem (1) directly, enforcing 99% of zeros. We found that the 319 features selected in such way led to a classification performance on the test set comparable to the one obtained with the set S_1 . Therefore, we conclude that the resampling strategy, indeed, allows to deal with a very large data matrix (possibly exceeding the available memory capacities) provided we require a gradual decrease of the number of selected features. A more drastic decrease in the number of features selected in each resampled problem lowers the overlap between the obtained solutions and

apparently yields a set S_1 which does not cover the whole spectrum of the relevant features. This effect, as discussed in the next Section, is enhanced by the high correlation between features which is typical for images and other autocorrelated signals.

B. A refinement of the solution

Before describing the two-stage solution we recommend, let us formulate some remarks about the peculiarities of the feature selection problem in computer vision.

1) *Feature selection in computer vision:* In general, the natural redundancy of image features – due to the specificity of the visual signal, rich of spatial (and temporal) correlation, does not compromise generalization performance. It may affect however the computational efficiency of the classifier. Because of such information redundancy, one could select one or few delegates from each group of correlated features to represent the other elements of the group. As for the choice of the delegate, unlike in other applications (such as microarray data analysis, for example), in most computer vision problems it would be good enough to select a random delegate in each group of correlated feature set. In other words, image features are not important *per se* but for the appearance information they carry, which is shared by all other members of the group.

At this point, we should remark that the face features we compute are correlated not only because of the intrinsic short range correlation of all natural images, or because they constitute an over-complete description, but also because of conditional dependencies induced by the class of face images (containing multiple occurrences of similar patterns at different locations, for example). We call these multiple occurrences, which are typical for the class of interest and usually occur on a longer range, *non trivial correlations*. Correlation due to the representation chosen produces redundant descriptions, while correlation due to the class of interest may carry important information on its specificities (see Figure 8).

2) *Iterating our selection scheme:* In order to obtain a smaller set of meaningful features we further apply our feature selection procedure to the set S_1 to get a new, sparser, solution. The new data matrix is obtained selecting from \mathbf{A} the columns corresponding to S_1 , and the initial iterate $\mathbf{f}_{S_1}^{(0)}$ is again a vector of zeros. Concerning the choice of the strategy for tuning the threshold parameter, we consider the combinations reported in Table II, our final choice is a combination of Method 1 for the first stage and Method 2 for the second. This choice is not driven by performance (the results obtained with the various methods are very similar – see Figure 9) but by two important issues: (A) Method 2 is computationally

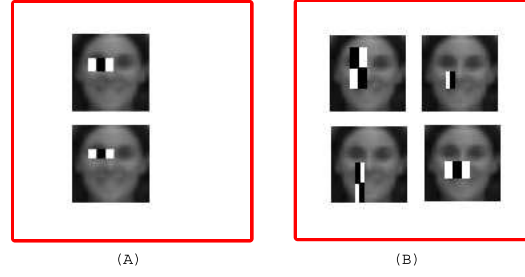


Fig. 8. On the left (A) a trivial spatial correlation, on the right (B) a group of four non-trivially correlated features, according to Spearman's correlation test.

| | II stage | |
|-----------------------------|----------|----------|
| I stage | method 1 | method 2 |
| method 1 (4636 features) | 231 | 247 |
| method 2 (1180 features) | - | 216 |

TABLE II
SIZE OF SELECTED FEATURE SETS WITH THE DIFFERENT PARAMETER TUNING METHODS.

expensive for the first stage (B) Method 2 takes explicitly into account generalization and therefore is more appropriate for the second stage. As concerns the choice of the parameter in the second stage, Table III shows how the average classification error and the number of features vary as a function of the parameter τ . We observe that the error is not too sensitive to the parameter choice, while the number of

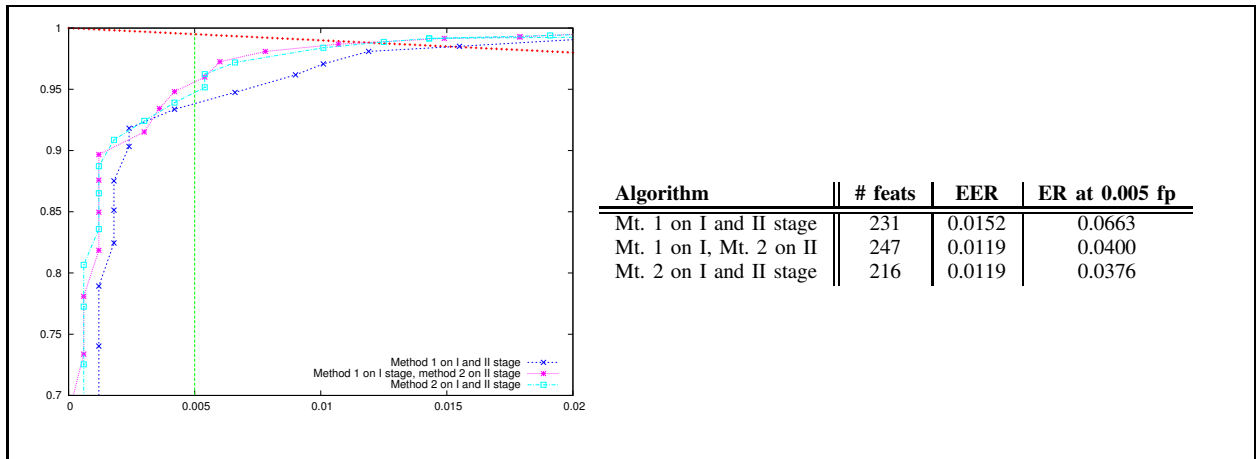


Fig. 9. Comparison between the methods for choosing τ . **Method 1** is based on setting the number of zeroes; **Method 2** is based on cross-validation (see text).

| τ | error | # of sel. feat. |
|--------------------|-------|-----------------|
| 1×10^{-8} | 0.022 | 4636 |
| 1×10^{-7} | 0.023 | 4593 |
| 1×10^{-6} | 0.022 | 2219 |
| 2×10^{-6} | 0.022 | 1922 |
| 5×10^{-6} | 0.023 | 1813 |
| 1×10^{-5} | 0.021 | 1791 |
| 2×10^{-5} | 0.020 | 1303 |
| 5×10^{-5} | 0.022 | 867 |
| 1×10^{-4} | 0.023 | 581 |
| 2×10^{-4} | 0.027 | 247 |
| 4×10^{-4} | 0.135 | 66 |
| 5×10^{-4} | - | 0 |

TABLE III

AVERAGE CLASSIFICATION ERROR AND NUMBER OF FEATURES FEATURES NUMBER AS A FUNCTION OF THE PARAMETER τ .

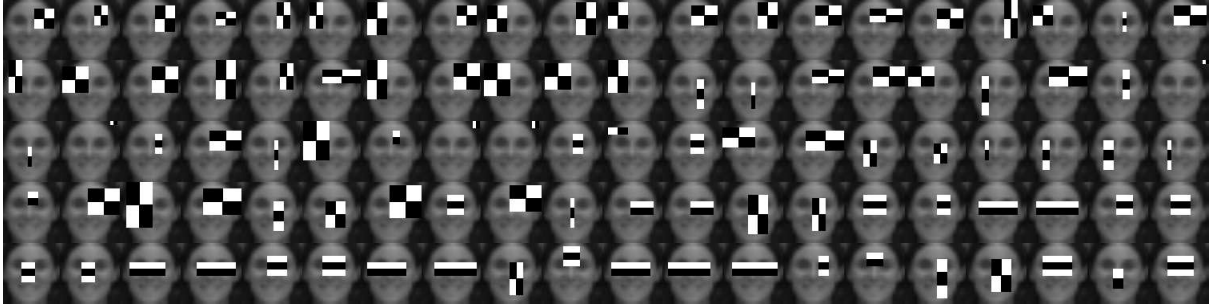


Fig. 10. The first 100 features out of the full set S_2 of 247 features obtained applying two regularized feature selection steps.

selected features depends critically from this choice. In this case we chose τ corresponding to an error lower than 0.1 and leading to the smaller set of features: $\tau = 2 \times 10^{-4}$.

3) *Experimental analysis:* As pointed out earlier, the generalization performances are similar for the different model selection methods under consideration, with nevertheless a slight advantage for the combination of Method 1 in the first and Method 2 in the second stage. Therefore in the following we stick to the latter choice.

Figure 9 reports the comparison between ROC curves and a table summarizing the results. The number of features kept at the end of this procedure is illustrated in Table II. Figure 10 shows the first 100 features out of the full set S_2 of 247 features; redundancy can still be noticed.

Figure 11 compares the classification results obtained by the proposed two-stage feature selection against PCA. Our set of features allows us to obtain higher classification rates and, moreover, only 247

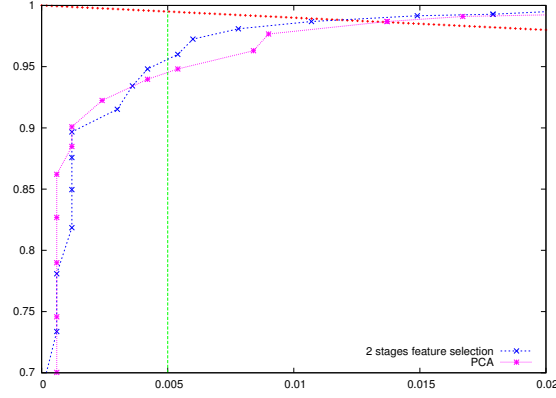


Fig. 11. A comparison of the classification results obtained applying a second regularized selection step or PCA to the set of features obtained from the first stage.

rectangle features per test need to be computed.

C. The final system

In this section we introduce a third, final optimization stage which further reduces the features by analyzing their empirical correlation. We then summarize the complete architecture of the proposed feature selection system and conclude with a comparative analysis with Adaboost feature selection.

1) *A third optimization stage:* The final set of features obtained with the two-stage method still contains groups of significantly overlapping features of the same type (see Figure 10). In order to obtain a reduced set of features with less redundancy, we devised a third selection stage where we select only one delegate from groups of trivially correlated features.

We design a selection method based on an analysis of the distance separating two features combined with Spearman's correlation test. We say that two features are (δ, ϵ) -uncorrelated if their distance is larger than δ or their Spearman's correlation coefficient [15] is smaller than ϵ . This approach allows us to discard similar features, while retaining non-trivial correlations.

We use a distance measure D between pairs of rectangle features f_i and f_j that takes into account both their shape and position and is defined as follows

$$D(f_i, f_j) = \begin{cases} \infty & \text{if } f_i \in \mathcal{F}_k \text{ } f_j \in \mathcal{F}_l \text{ } k, l = \{2h, 2v, 3h, 3v, 4\} \text{ } k \neq l \\ d(f_i, f_j) & \text{otherwise} \end{cases}$$

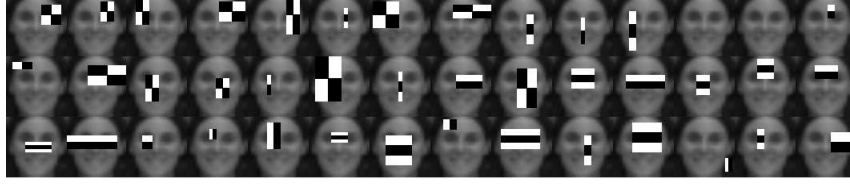


Fig. 12. The 42 features that are left after the third selection stage (see text) applied to the features shown in Fig. 10.

where $d(f_i, f_j)$ is the sum of the Euclidean distances between the corresponding corners of the rectangular supports of the two features. Accordingly, the distance between two features of different type is infinite, while the distance between features of the same type increases with changes of position, scale and aspect ratio. Clearly, this distance is specially designed for the rectangle features but the definition could be easily adapted for other local descriptions.

We start from the set S_2 , with the features ranked according to their weight and obtain a reduced set of features S_3 as follows. Starting from the top of the list, we examine each feature f_j in S_2 , but include it in S_3 only if f_j is (δ, ϵ) -uncorrelated with respect to all other features already in S_3 .

Using this further selection stage we obtain a very compact description set with little degradation of the results. Figure 12 shows the 42 features selected by this third stage: the representation is compact and descriptive of the various areas on a face with very little overlapping. The classification results are still satisfactory, presenting only a small degradation with respect to the performances of the set selected through the two-stage procedure. Figure 13 compares the ROC curves obtained with the two-stage feature selection, with and without the additional third-stage selection.

2) *The final three-stage architecture:* The discussion reported in the previous sections leads us to adopt a three-stage feature selection strategy which is summarized by Figure 14. In the first stage we start from the whole set of features and extract a first subset S_1 running S times the iterative thresholding algorithm (Section IV-A). In the second stage the feature selection method is applied again on S_1 (Section IV-B), and a reduced set of features S_2 is obtained. As for model selection (i.e., the procedure to tune the parameter τ in the various stages) we apply Method 1 in the first and Method 2 in the second stage (the two methods are discussed in Section IV-A and their combination in Section IV-B).

Finally the third stage relying on a correlation analysis allows us to obtain a very small set of features S_3 where most of the redundancy has been eliminated.

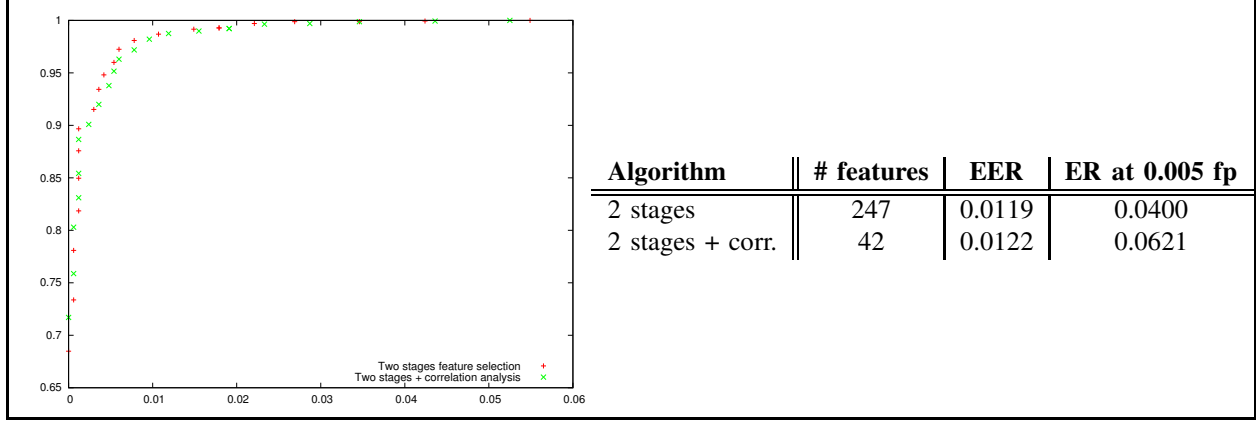


Fig. 13. Two-stage feature selection with and without the additional third-stage selection based on a correlation analysis. The results show that the third stage allows a considerable reduction of the set size with little degradation of the performance.

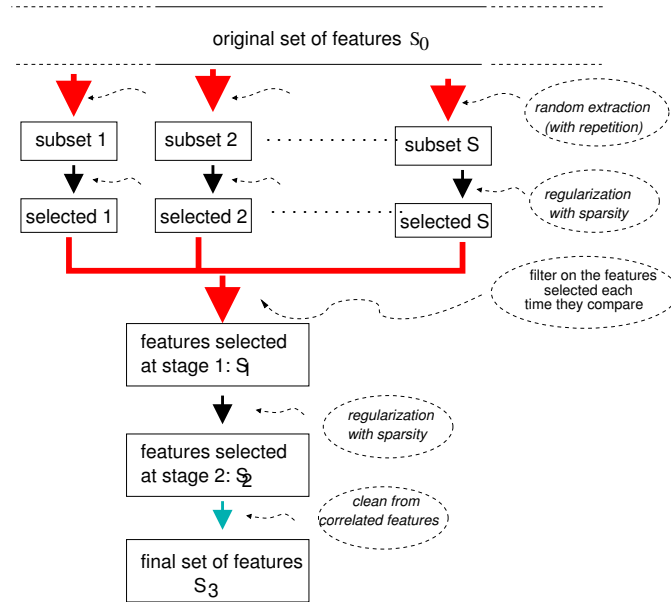


Fig. 14. The flow of feature sets from the original description to the final set (dashed balloons indicate the different filtering operations performed).

3) *Comparative analysis*: A strength of our method is that a relative small training set appears to be sufficient for obtaining a very good set of selected features. In the first part of these experiments, we used the same number of positive and negative examples for the training, namely 2000 positive and 2000 negative examples. State-of-the-art approaches [33] often use a huge amount of examples (from tenths of thousands to millions), in particular of negative examples, to model the high variability of this

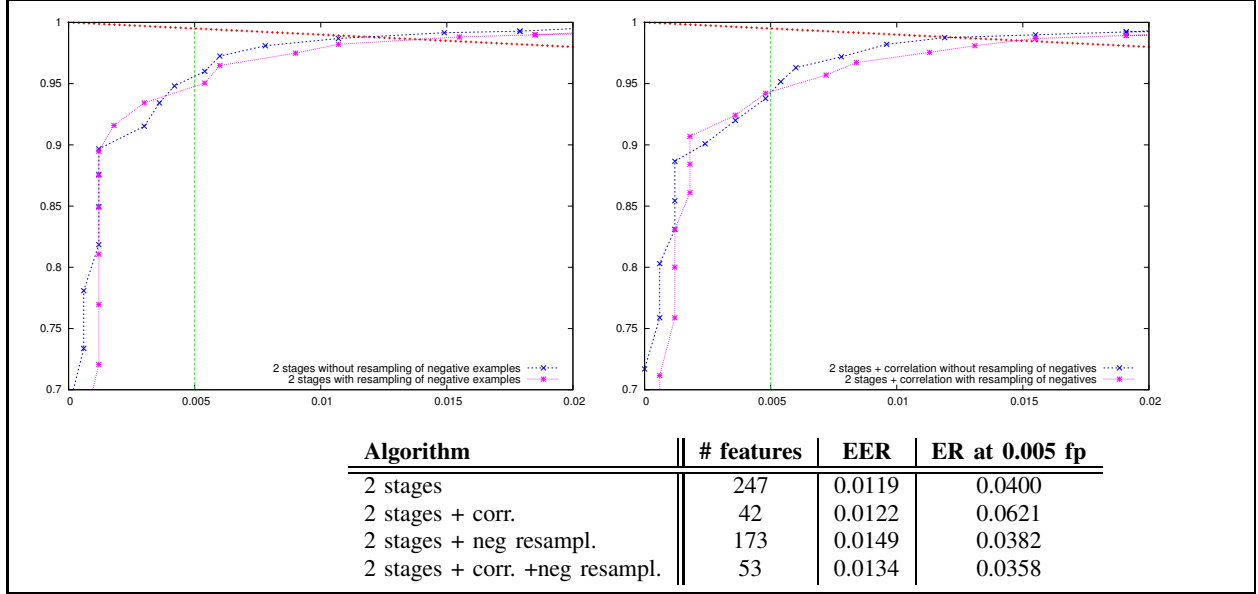


Fig. 15. Comparative analysis with respect to generalization performance between our method applied to the original dataset (2000 positive and 2000 negative examples respectively) and the same method with a higher set of negative examples (a total of 20000 negative examples).

latter class. Hence, one may ask whether our results could be improved by using a bigger set of training examples. Since, in our case, adding more data would mean dealing with a bigger feature matrix, we exploit once more the resampling strategy: at each resampling of the features, we also resample 2000 new negative examples from a pool of 20000 examples.

Figure 15 compares the results obtained with the first two-stage selection, with and without resampling of negative examples (in the absence resampling only the first 2000 negative data of the total pool are used). Using a higher number of negatives does not seem to improve the results neither on the two-stage (see Figure 15 left) nor on the three-stage selection (Figure 15 right). Hence our method does not seem to benefit from a bigger set of negative examples.

Even if our paper is focused on feature selection rather than on the design of the classifier, we have also considered other kernel functions for the SVM. In particular we have implemented a *polynomial* kernel of degree 2, well known for capturing short-range correlations. Figure 16 compares linear and polynomial SVMs on data representations based on two different selection stages: on the left the results obtained representing data with features in S_2 (set obtained after 2 stages) and on the right the results obtained with features in S_3 (set obtained after 3 stages). It can be observed that, if the features are correlated,

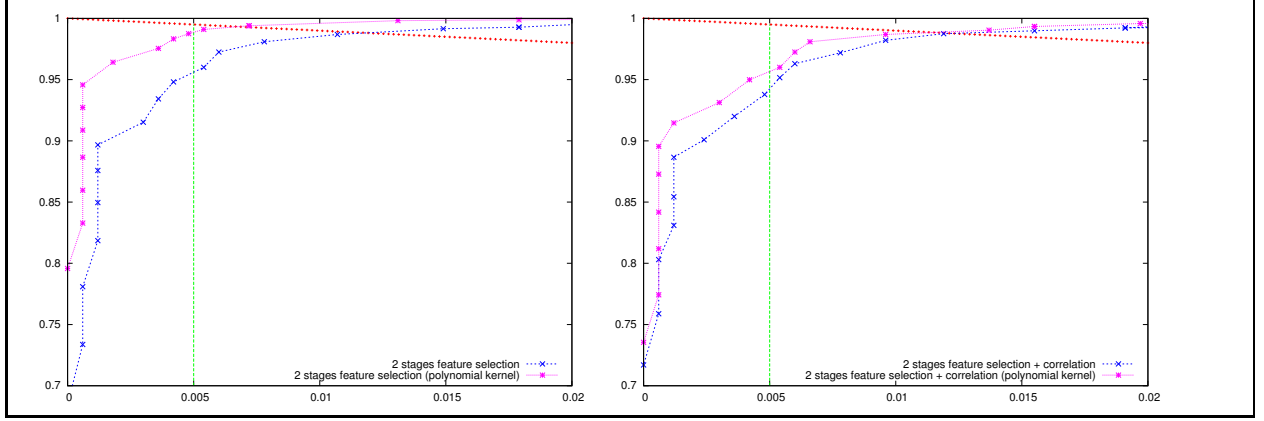


Fig. 16. Comparison between linear and polynomial kernels. Left: results obtained representing images with the set S_2 (247 features from the two-stage architecture); right: results obtained with the set S_3 (42 features from the three-stage architecture). It is seen that in the absence of correlation between features there is no gain in using polynomial kernels.

there is an advantage in using polynomial kernels, but, if not, the results obtained are equivalent. Notice that the computational cost is smaller when using the linear kernel. Therefore, whenever fast computation is required, it is advisable to compute the set S_3 and choose a linear kernel.

We compared the generalization performance obtained with our three-stage feature selection method with the ones obtained with the Adaboost feature selection proposed in [33]¹. This comparison focuses on the quality of the selected features: again, we evaluate the goodness of features with respect to their generalization ability, then in both cases we check how good they are on a classification task, using a linear SVM classifier. Figure 17 shows a comparison between classification performances obtained with our features (with and without the third stage of correlation analysis) and with the Adaboost features selected from *our same set of data*. Since in [33] feature selection and training are performed on a unique Adaboost loop, we checked the fairness of our comparison by including on our ROC plots the false positive / hit ratio obtained with a cascade of Adaboost classifiers (marked as a black dot). Notice that these results are in line with the curve obtained with Adaboost features. We conclude by remarking that the impressive performances obtained by the face detection method in [33] seem to rely on the use of a very big set of negative examples. At each stage of the cascade the current classifier is trained with new negative examples, while the ones that were correctly classified in the previous stages are discarded.

¹To carry on these experiments we used the implementation of the Adaboost face detector available with the Intel OpenCV libraries <http://http://www.intel.com/technology/computing/opencv/>.

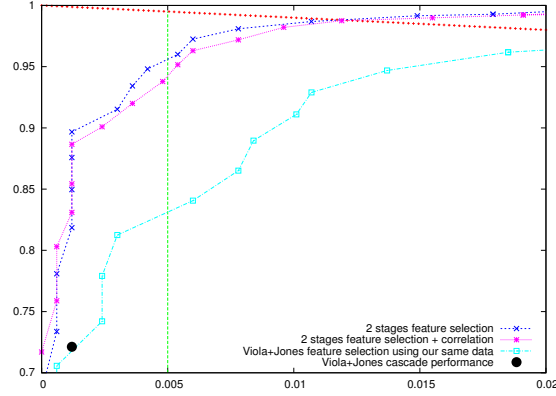


Fig. 17. Comparing our approach with Adaboost features using our same pool of data

V. A FULLY TRAINABLE SYSTEM FOR DETECTING FACES AND EYES

In this section we apply our feature selection method to face and eye detection. In both problems, as in most object detection problems, the majority of the analyzed image patches are just background (i.e. negative examples). Also, large portions of the background areas are very different from the face pattern, therefore they are “easy” negative examples. We therefore adopt a coarse-to-fine architecture, namely a *cascade* of classifiers, as often used to speed up detection without losing too much in terms of generalization performances [1], [6], [12], [33]. Interestingly, as pointed out in [33], a judicious use of cascading may lead to a better detector than the use of a monolithic classifier.

A. The cascading architecture

Our mechanism to build the cascade works on the output of our three-stage feature selection scheme, that is on the set of features S_3 ordered according to the weights assigned to features in the second stage.

Our aim is to build many SVM classifiers (that we may call *weak* classifiers if adopting the Adaboost terminology) trained on the same set of data each time represented according to a different subset of S_3 .

Let us briefly discuss how we compute the features subsets. Each subset contains *at least* 3 distant features that are able to reach a fixed target performance on a validation set. We start by 3 mutually distant features (according to distance measure (5)) picked up from S_3 , train a SVM classifier, check whether it reaches a target performance on the validation set. If it does not, add further (distant) features from S_3

until performance is reached. Then, if S_3 is not empty, start with a new set of 3 distant features to build a new subset. At the end of this procedure we are left with a list of T very small (and therefore efficient) linear SVM classifiers, each one describing different aspects of the object of interest. The cascade of classifiers is built rather conventionally. The aim is that a positive (face or eye region) is detected only if all the classifiers give a positive feedback. Instead, a region is discarded as soon as a classifier gives a negative answer.

Target performance is chosen so that each classifier is unlikely to miss positives: we set the minimum hit rate to 99.5% and the maximum false positive rate to 50%. As pointed out by Viola and Jones [33], such a modest target allows us to achieve good performances with the global classifier, since the global false positive rate F and the global hit rate H can be computed as

$$Fp = \prod_{i=1}^K (fp)_i \quad \text{and} \quad H = \prod_{i=1}^K h_i,$$

where $(fp)_i$ is the false positive rate and h_i is the hit rate for each classifier i . With a cascade of 10 classifiers, we would get $H = 0.995^{10} \sim 0.9$ and $Fp = 0.5^{10} \sim 3 \times 10^{-5}$.

At the end of the cascade design, we test the system on live videos and check whether the results meet our performance needs. If not, we repeat the second and third selection stages on a new set of negative examples made of all the false positives detected by the face detector in this validation phase (this approach is similar to the bootstrap procedure adopted by many authors, see for instance [20]). The features we obtain from this new selection are more specialized in discriminating between faces and difficult negative examples. The classifiers obtained from this refinement are added at the bottom of the cascade.

B. Face detection module

To do face detection, we build up a cascade of classifiers starting from the 42 features obtained in Section IV-C (see Figure 12). The cascade we obtain is made of 13 layers each of which made of at most 4 features; this means that in the construction process 4 features were always enough to meet the target performances. Unlike other cascades of classifiers, in this case all stages are composed by very small classifiers.

The performance we obtained applying the cascade of SVMs on our test set of faces and non-faces is 0.1% false positives and 94% hit rate.

Then we apply our cascade of SVMs to new video data acquired from our monitoring system. Image processing, including image normalization and multi-scale scanning, follows the suggestions made in [33]: our face detector analyzes each frame looking for faces in the image area where motion was detected. Since faces of different size could appear, a multi-scale search is performed; we implement it with a rescaling of the features rather of the image. We then test all locations shifting the search window of one pixel at a time. Given an image patch at a certain location and scale we use it as input for the classifiers cascade. Since multiple detections should occur in the surroundings of a face (because of the spatial correlation of the image), we discard areas with few hits (less than 4 in our experiments). Empirically, we observe that the performance of our face detector is satisfactory. In order to report a quantitative evaluation we labelled a 5 hours recording from our monitoring system. Figure 19 shows sample frames processed by our face detector where the detected faces (and eyes, see ahead in this section) are highlighted. The results obtained on this 5 hours recording are of about 4×10^{-7} false positives, and a hit rate of 76%. Temporal coherence between subsequent frames was not exploited in the experiments reported in this work.

C. Eye detection module

Our automatic feature selection procedure, being entirely data-driven, can be applied to other datasets or other high-dimensional descriptions to extract the most discriminative features for the problem at hand. To confirm this we apply the same strategy to a problem of *eye detection*.

Instead of manually labelling images from our system, we collect a dataset of 1300 positive and 1300 negative image patches containing the eye region²: we used 2000 images as training set and the remaining as validation set. Notice that, again, the size of the training and validation set is quite small. The size of these regions is 40×20 pixels, thus we generate a higher number of features than in the case of faces: we start from 243800, keeping 20574 after the first stage, 1372 after the second. At the end of the third stage we are left with 82 features (see Figure 18). The cascade structure we obtain is made of 22 layers.

For a quantitative experimental evaluation of the eye-detector run as a stand-alone module, the reader is referred to the next section. Here we comment on the performances of the eye detector currently installed

²The dataset is a portion of the Feret database <http://www.nist.gov/humanid/feret>. We used ground truth information on the position of face features to automatically extract the eye regions.

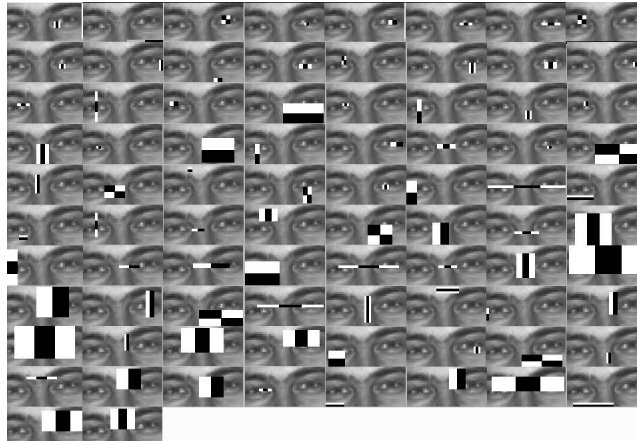


Fig. 18. The 82 eye features automatically extracted with our three-stage method. Notice that a majority of features is localized on the eyes while a smaller number of features capture the overall structure of the eyes region.

on our monitoring system, and running on the output of face detection, in the case the detected face is bigger than 40×40 pixels — twice the minimum eye region size.

We observe a slight decrease of both false positive rate and hit rate, but, at the same time, we see how the quality of the detected faces improves — in particular the combined face+eye detection limits the amount of non-frontal faces. The quality of the detected faces is important for the next step of our system pipeline: a face authentication module which is currently under development. In this module the eye region is also used to automatically register faces, that is, as a reference to align images so that the corresponding semantic features (eyes, nose, ...) appear in similar positions. Preliminary results are convincing and speak for the localization robustness of the method. Figure 19 reports results on the combined face and eye detection.

D. Experiments on benchmark datasets

To assess the robustness of our method to changes of dataset, we tested it on the test sets A and C of the MIT-CMU dataset for faces³ and on the BioId face and eyes dataset⁴.

As concerns face detection we used 4000 images from the CBCL training set for frontal faces as training and validation set, since the data we acquire with our system are too specific to our application

³Available at http://vasc.ri.cmu.edu/idb/html/face/frontal_images/.

⁴Available at <http://www.bioid.com/downloads/facedb>.



Fig. 19. Face and eye detection on sample frames acquired by our monitoring system: detected faces are marked in green, detected eye regions in blue. The minimum face size is of 19×19 pixels, while the minimum eye region size is of 20×40 pixels.

scenario to be used for any kind of images. Since the variety of negative data is higher than in our case study, we applied the bootstrapping procedure. We obtain two separate sets of features, of size 37 and 59 respectively, from which we compute a single cascade with 19 layers: 11 classifiers obtained from the first set of features, 8 obtained from the second set. Since the latter are computed from a more difficult set of data, each classifier uses bigger set of features, of size 8 on average. It is interesting to see how the features selected from the CBCL dataset (see Figure 20) are different from the ones selected on our dataset, and how they capture the vertical symmetry that is not present in the images acquired with our system, where training images were not exactly frontal.

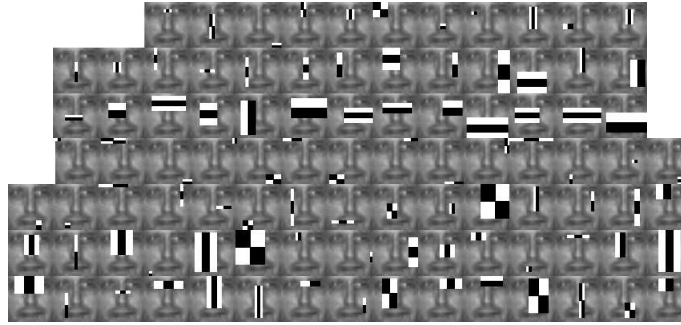


Fig. 20. The features selected from the CBCL dataset, before (top) and after (bottom) the bootstrapping procedure.

Figure 21 reports detections on sample images of the dataset. The results obtained are very good even if the training set is relatively small. We observed an increase of performance when applying our cascade with respect to the cascade produced by Adaboost feature selection trained on the same training set (4000 images). Indeed, keeping the false positive rate fixed, the hit-rate gain with our approach is of about 8%. We also run the Adaboost face detector on a selection of features available among the OpenCV demos. In this case we do not have precise information on the size of the training set but, from the preparation process described along with the demo, we can guess that it consists of millions of negative images (automatically cropped from a high number of random images not containing faces). The results obtained with this fully performing Adaboost are above our performance of about 10%. As concerns eye detection we adopt the same features as those selected from the Feret dataset. We do not apply the bootstrap process, even though we feel that it might improve the results of the eye-detector if used stand-alone.

As for test sets we choose the BioID dataset, made of about 1500 images, each one containing one face under different illumination and variable pose expression. The dataset includes a ground truth on the position of face and eyes, making it easy to evaluate performances.

The images in BioID are more challenging for eye detection than our application setting, in the sense that very different conditions are depicted. Also, one third of images depict people wearing glasses in neon-illuminated environments (therefore many of them suffer from the presence of reflections). Concerning the glasses, a remark is in order: in our setting, faces (and eye pairs) are quite small due to the relative distance between the people and the camera. Therefore the impact of glasses is usually

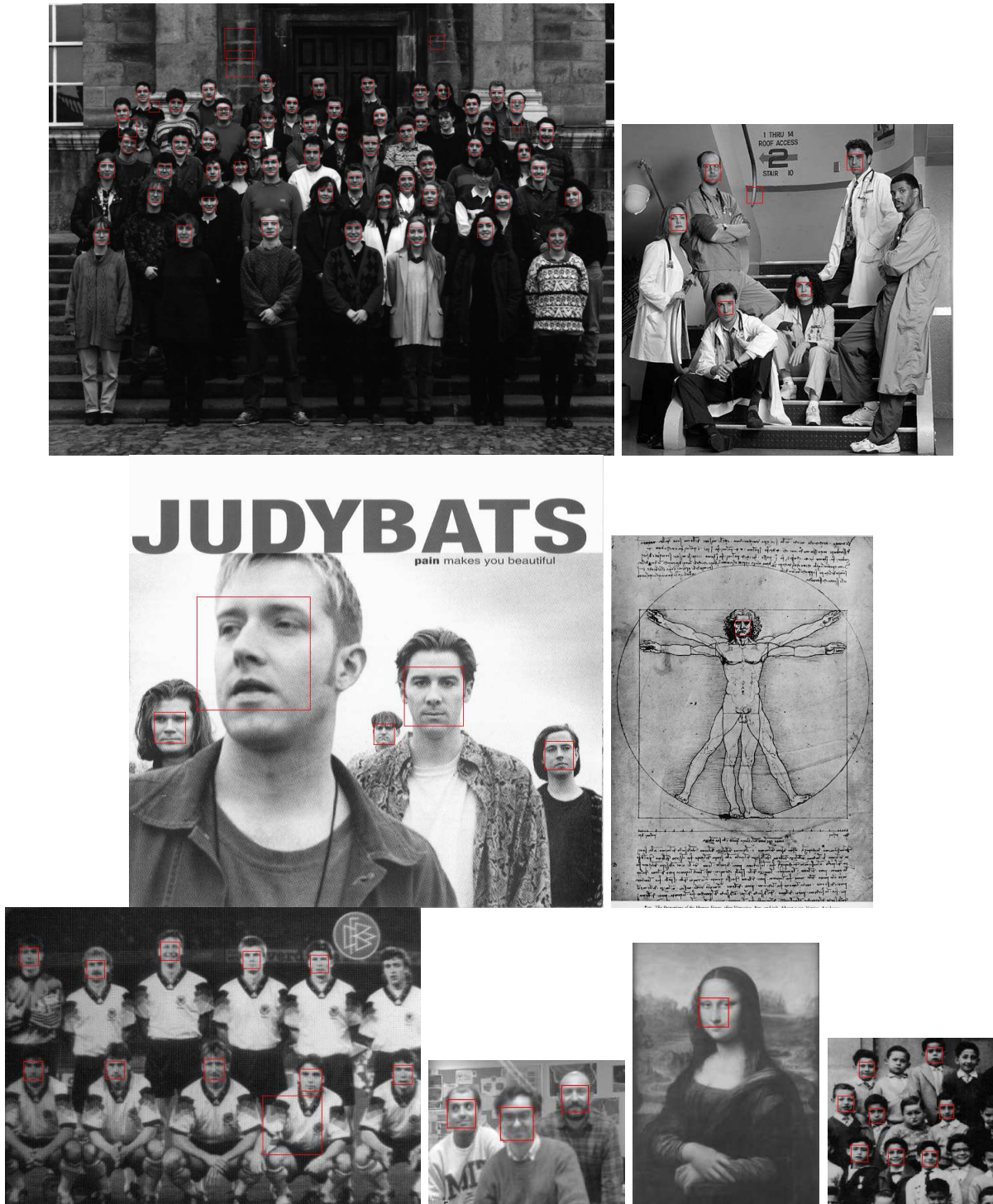


Fig. 21. Face detection results on example images from the CMU dataset

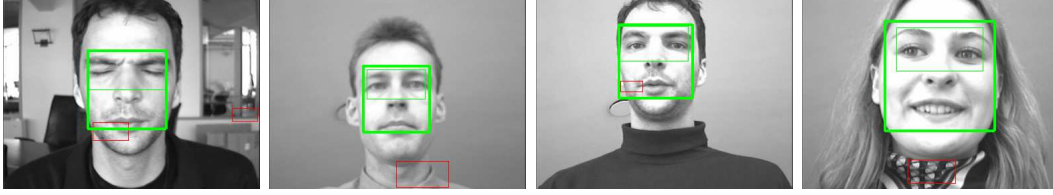


Fig. 22. Face and eye detection results on sample frames from the BioID dataset. True positives are green, and false positives are red.

negligible. This is not the case with BioID images. Indeed, the performance of the eye detector reaches a 70% hit rate with a false positive rate of $3.29 \times 10^{-7}\%$ on the whole dataset (that is, we obtain 3631 false positives on all the tests performed on 1520 images of size 384×286 , each one containing one face only), while the hit rate goes up to 86% with a false positive rate of $3.26 \times 10^{-7}\%$ if the dataset is cleaned from images with reflections on the glasses (this time we considered 1029 images and obtained 2437 false positives). Figure 22 shows face and eye detection results on samples from the BioID. Green rectangles mark true positives (estimated according to the ground truth), and red rectangles false positives. The figure suggests how a combination of face and eyes may improve performances.

VI. DISCUSSION

This present work is built upon an iterative algorithm implementing the so-called Lasso scheme as a feature selection procedure. We explored its effectiveness in the field of computer vision, taking into account the peculiarities of image features — strongly affected by the spatial correlation of images.

The global selection strategy we devised includes three stages. In the *first* stage, for computational reasons, we proposed to split a big problem into a number of smaller problems involving randomized blocks of features in order to select a smaller but still highly redundant set of features; this stage could be further optimized by exploiting parallel computation. The *second* stage is based on applying again the selection algorithm to the features resulting from the first stage; the obtained set is, on average, 0.5% of the size of the initial representation: we experimentally validated its appropriateness on a classification task, obtaining very good results. The *third* stage is based on finding a very small set of uncorrelated features that is suitable for real-time processing to the price of a very limited decrease in performance. We tested the latter description on a real-time face and eye detection system based on a cascade of SVM classifiers. This system is currently running as a module of a prototype monitoring system developed in

our laboratory. For completeness we tested our real-time face detector on publicly available data obtaining comparable results to state-of-the-art approaches trained on the same dataset.

From the algorithmic point of view, the main advantage of our approach is that it is trained on a very small training set (compared with other state-of-the-art methods). This is not only advantageous to facilitate the data-gathering stage, but, more importantly, to limit the training time. Indeed, we empirically observed that our feature selection procedure requires a training time comparable to Adaboost methods trained on the same dataset (about 20 hours). Then, achieving state-of-the-art performance with a smaller training set is a clear advantage. As for the test phase, notice that we perform fewer operations per evaluation (one summation and one thresholding to compute the SVM output) than Adaboost (one thresholding per feature — weak classifier plus one summation plus the final thresholding). Considering the high number of evaluations performed per image such saving will have incidence on the overall performance. Let us stress again the fact that our resampling strategy, based on finding the solutions of many subproblems, can be naturally cast in a parallel computation framework.

We are currently working on a speed-up procedure that will allow us to further reduce the training time. Moreover, we are developing a face authentication module that will constitute a further demonstration of the versatility of the method we propose.

Acknowledgments

We would like to thank Lorenzo Rosasco and Ernesto De Vito for many useful and stimulating discussions. Also we thank the reviewers of the manuscript for their constructive comments and useful insights. This work has been partially supported by the FIRB Project LEAP (RBINO4PARL). Christine De Mol acknowledges the hospitality of DISI during a sabbatical semester in Genoa and the support of the grants ARC 02/07-281 and GOA62 (VUB).

REFERENCES

- [1] Y. Amit and D. Geman. A computational model for visual selection. *Neural Computation*, 11:1691–1715, 1999.
- [2] P. J. Burt. Smart sensing within a pyramidal vision machine. *Proc of the IEEE*, 76(8), 1988.
- [3] S. S. Chen, D. Donoho, and M. Saunders. Atomic decomposition by basis pursuit. *SIAM Journal on Scientific Computing*, 20(1), 1998.
- [4] I. Daubechies, M. Defrise, and C. De Mol. An iterative thresholding algorithm for linear inverse problems with a sparsity constraint. *Comm. on Pure Appl. Math.*, 57:1416–1457, 2004.

- [5] A. Destrero, C. De Mol, F. Odone, and A. Verri. A regularized approach to feature selection for face detection. In Y. Yagi et al., editor, *Proc. of the Asian Conference on Computer Vision, ACCV*, LNCS 4844, pages 881–890, 2007.
- [6] F. Fleuret and D. Geman. Coarse-to-fine face detection. *International Journal on Computer Vision*, 41:85–107, 2001.
- [7] J. Fowler. The redundant discrete wavelet transform and additive noise. *IEEE Signal Processing Letters*, 12, 2005.
- [8] E. Franceschi, F. Odone, F. Smeraldi, and A. Verri. Feature selection with non-parametric statistics. In *IEEE Intern. Conference on Image Processing*, 2005.
- [9] Y. Freund and R. E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. In *European Conference on Computational Learning Theory*, pages 23–37, 1995.
- [10] J. Friedman, T. Hastie, and R. Tibshirani. Additive logistic regression: a statistical view of boosting, 1998.
- [11] I. Guyon and E. Elisseeff. An introduction to variable and feature selection. *Journal of Machine Learning Research*, 3:1157–1182, 2003.
- [12] B. Heisele, T. Serre, S. Mukherjee, and T. Poggio. Feature reduction and hierarchy of classifiers for fast object detection in video images. In *IEEE Proc. CVPR*, 2001.
- [13] B. Heisele, T. Serre, M. Pontil, and T. Poggio. Component-based face detection. In *CVPR*, 2001.
- [14] L. Itti, C. Koch, and E. Niebur. A model of saliency-based visual attention for rapid scene analysis. *IEEE Trans. on Pattern Analysis and Machine Int*, 20(11), 1998.
- [15] E. L. Lehmann. *Nonparametrics: Statistical methods based on ranks*. Holden-Day, 1975.
- [16] Stan Z. Li and ZhenQiu Zhang. FloatBoost learning and statistical face detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(9), 2004.
- [17] S. Mann and S. Haykin. The chirplet transform: A generalization of Gabor’s logon transform. *Vision Interface ’91*, pages 205–212, 1991.
- [18] A. Mohan, C. Papageorgiou, and T. Poggio. Example-based object detection in images by components. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 23(4):349–361, 2001.
- [19] B. A. Olshausen and D. J. Field. Sparse coding with an overcomplete basis set: a strategy employed by V1? *Vision Research*, 37(23):3311–3325, 1997.
- [20] E. Osuna, R. Freund, and F. Girosi. Training support vector machines: an application to face detection. In *CVPR*, 1997.
- [21] C. Papageorgiou and T. Poggio. A trainable system for object detection. *International Journal of Computer Vision*, 38(1):15–33, 2000.
- [22] M. Pontil and A. Verri. Support vector machines for 3-d object recognition. *IEEE Trans. on PAMI*, 20:637–646, 1998.
- [23] N. Rajpoot, R. G. Wilson, F. G. Meyer, and R. Coifman. A new basis selection paradigm for wavelet packet image coding. In *IEEE Int. Conf. on Image Processing*, 2001.
- [24] A. Resefield and G. J. Vanderbrug. Coarse-fine template matching. *IEEE Trans. on Sys. Man. Cyb.*, 2, 19677.
- [25] D. Roth, M. Yang, and N. Ahuja. A snowbased face detector. *Neural Information Processing*, 12, 2000.
- [26] H. Schneiderman and T. Kanade. A statistical method for 3D object detection applied to faces and cars. In *International Conference on Computer Vision*, 2000.
- [27] F. Smeraldi. Ranklets: orientation selective non-parametric features applied to face detection. In *Proc. of Intern. Conference on Pattern Recognition*, volume 3, pages 379–382, 2002.

- [28] K. Sung and T. Poggio. Example-based learning for view-based face detection. *IEEE Trans. on Pattern Analysis and Machine Int.*, 20:39–51, 1998.
- [29] R. Tibshirani. Regression shrinkage and selection via the lasso. *J Royal. Statist. Soc. B*, 58(1):267–288, 1996.
- [30] J. K. Tsotsos, S. M. Culhane, W. Y. Wai, Y. H. Lai, N. Davis, and F. Nuflo. Modeling visual attention via selective tuning. *Artificial Intelligence Journal*, 78(1-2), 1995.
- [31] V. N. Vapnik. *The nature of statistical learning theory*. Springer–Verlag, 1995.
- [32] V. N. Vapnik. *Statistical Learning Theory*. Wiley, 1998.
- [33] P. Viola and M. J. Jones. Robust real-time face detection. *International Journal on Computer Vision*, 57(2):137–154, 2004.
- [34] Q. Wang, J. Yang, and W. Yang. Face detection using rectangle features and SVM. *Int. Journ. of Intelligent Technology*, 1(3), 2006.
- [35] J. Weston, A. Elisseeff, B. Scholkopf, and M. Tipping. The use of zero-norm with linear models and kernel methods. *Journal of Machine Learning Research*, 3, 2003.
- [36] M.-H. Yang, D. J. Kriegman, and N. Ahuja. Detecting faces in images: a survey. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 24(1):34–58, 2002.
- [37] D. Zhang, S. Z. Li, and D. Gatica-Perez. Real-time face detection using boosting in hierarchical feature spaces. In *Proc. of ICPR*, 2004.
- [38] J. Zhu, S. Rosset, T. Hastie, and R. Tibshirani. 1-norm support vector machines. In *Advances in Neural Information Processing Systems 16*. MIT Press, 2004.