ABSTRACT

Title of dissertation: LOOKING AT PEOPLE USING PARTIAL LEAST SQUARES

William Robson Schwartz
Doctor of Philosophy, 2010

Dissertation directed by: Professor Larry S. Davis

Analysis of images involving humans is of significant interest in computer vision because problems such as detection, modeling, recognition, and tracking are fundamental to model interactions between people and understand high-level activities. Visual information contained in images is generally represented using descriptors (features). Many general classes of descriptors have been proposed focusing on different characteristics of images. Therefore, if one considers only a single descriptor, one might ignore useful information for a given task, compromising performance. In this research we consider a rich set of image descriptors analyzed by a statistical technique known as Partial Least Squares (PLS). PLS is a class of methods for modeling relations between sets of observations by means of latent variables and it is used to project exemplars from a very high dimensional feature space onto a low dimensional subspace. We demonstrate the effectiveness of combining a richer set of descriptors using PLS in two significant tasks in computer vision. First, we propose

a method to detect humans, which is then extended to handle partial occlusion and finally a framework based on PLS regression models is incorporated to further reduce the computational cost. Second, an object recognition framework based on a one-against-all scheme is exploited for appearance-based person modeling and face identification.

LOOKING AT PEOPLE USING PARTIAL LEAST SQUARES

by

William Robson Schwartz

Dissertation submitted to the Faculty of the Graduate School of the
University of Maryland, College Park in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
2010

Advisory Committee:
Professor Larry S. Davis, Chair/Advisor
Professor Amitabh Varshney
Professor David Jacobs
Professor Min Wu
Professor Ramani Duraiswami

Dedication

To Homéro and Maria

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

Analysis of images involving humans (application domain known as *looking at people* [22]), is of significant interest in computer vision because problems such as detection, modeling, recognition, and tracking are fundamental to model interactions between people and understand high-level activities.

Image descriptors (features) are generally used to extract and represent visual information contained in images. Many general classes of descriptors have been proposed focusing on different characteristics of images [13, 49, 39, 8, 25]. Due to that, if one considers only a single descriptor, one might ignore useful information for a given task, compromising performance. Therefore, the use of a strong set of descriptors is desirable.

Combinations of low-level feature descriptors have provided improvements in tasks such as detection and recognition. A strong set of features provides high discriminatory power, often reducing the need for complex classification methods. Improvements in human detection have been achieved by using combinations of low-level features [11, 78]. Several types of missclassifications can be largely avoided once information such as homogeneity inside the body and difference between background and foreground regions is considered. In face recognition, works such as [67, 85] also have obtained improved results by combining multiple feature channels, particularly

when data collected under uncontrolled conditions is considered.

A consequence of feature augmentation is an extremely high dimensional feature space, rendering many classical machine learning techniques intractable. Additionally, the number of positive samples in the training dataset is much smaller than the number of dimensions. Furthermore, to obtain better discrimination, features need to be extracted from neighboring blocks within a detection window, which increases the multicollinearity of the feature set. Therefore, the nature of this feature set makes an ideal setting for Partial Least Squares (PLS) [75]. PLS is a class of methods for modeling relations between sets of observations by means of latent variables and it is used to project a very high dimensional feature space onto a low dimensional subspace.

Although originally proposed as a regression technique, PLS can be also be used as a *class aware* dimensionality reduction tool. We use PLS to project our high dimensional feature vectors onto a low dimensional subspace. In such low dimensional spaces, standard machine learning techniques such as linear and quadratic classifiers, SVMs, and k-nearest neighbors can be applied to perform classification. In addition, we exploit PLS regression as a way of feature weighting to perform one-against-all classification for object recognition applications.

In this research we consider a rich set of image descriptors analyzed by partial least squares (PLS). We show the effectiveness of combining a richer set of descriptors using PLS in two significant tasks in computer vision. First we propose a method to detect humans based on PLS, which is then extended to handle partial occlusion and finally a framework based on PLS regression models is incorporated to further

reduce the computational cost. Second, an object recognition framework based on a one-against-all scheme is exploited for appearance-based person modeling and face identification.

1.1 Human Detection

Effective techniques for human detection are of special interest in computer vision since many applications involve people's locations and movements. Over the last few years the problem of detecting humans in single images has received considerable interest. Variations in illumination, shadows, and pose, as well as frequent inter- and intra-person occlusion render this a challenging task.

To overcome problems faced in human detection we propose an approach that augments widely used edge-based features with texture and color information, providing us with a much richer descriptor set. Then, the approach is extended by combining information from face detection to handle partial occlusions. Finally, a set of regression models is integrated with the detector to reduce the computational cost.

Humans in standing positions have distinguishing characteristics. First, strong vertical edges are present along the boundaries of the body. Second, clothing is generally uniform. Clothing textures are different from natural textures observed outside of the body due to constraints on the manufacturing of printed cloth. Third, the ground is composed mostly of uniform textures. Finally, discriminatory color information is found in the face/head regions. In chapter 2 we exploit feature

augmentation analysed by PLS to improve detection accuracy. This method is referred to as *the PLS detector*.

Human detection under occlusion is also a challenging problem in computer vision. In chapter 3, we address this problem through a framework which integrates face detection and person detection. We first investigate how the response of a face detector is correlated with the response of a person detector. From these observations, we formulate hypotheses that capture the intuitive feedback between the responses of face and person detectors and use it to verify if the individual detectors' outputs are true or false.

The combination of multiple feature channels allows the PLS detector to be reliably used in different scenarios. However, even though this approach provides accurate detection results, as it will be shown in chapter 2, it leads to a high computational cost. On the other hand, if characteristics of the scene are known beforehand, a set of simple and fast computable features might be sufficient to provide high accuracy at a low computational cost.

Therefore, it is valuable to seek a balance between these two extremes such that the detection method not only works well in different scenarios but also is able to extract enough information from a scene to reduce the computation cost. With this purpose, in chapter 4 we integrate a set of data-driven regression models with the PLS detector to reduce the computational cost.

Experiments show that the use of multiple feature channels combined by PLS provides detection results that outperform state-of-art approaches on multiple standard datasets. In addition, the integration of person and face detectors improves

human detection under occlusion, as well as reduces the number of false alarms. Finally, the incorporation of data-driven regression models with the PLS detector provides significant speed-up with a slight improvement in detection accuracy, which is an important step towards achieving real time detection without loosing accuracy.

1.2 Appearance-Based Object Modeling and Recognition

Appearance-based modeling plays an important role when detection is performed in video. A human detector may not be able to perform well in every frame due to scale and pose variations present in the videos. Therefore, one might consider building appearance-based models for the detected people, and then use these models when the human detector fails. However, one of the main problems of using appearance-based discriminative models is the ambiguities among classes when the number of persons being considered increases. To reduce the amount of ambiguity, we propose the use of a rich set of feature descriptors based on color, textures and edges.

As well as in human detection, the nature of the input data poses great challenges to appearance-based modeling and the use of a single feature channel, such as color-based features, may not be powerful enough to capture subtle differences between different people's appearances. Therefore, additional cues need to be exploited and combined to improve discriminability of appearance-based models. In chapter 5 we describe a one-against-all scheme to build discriminative models using PLS to weight the features according to their discriminative power for each different

appearance.

The projection vectors estimated by PLS provide information regarding the importance of features as a function of location. Since PLS is a class-aware dimensionality reduction technique, the importance of features in a given location is related to the discriminability between appearances. High weights are located in regions that better distinguish a specific appearance from the remaining ones. This characteristic and the reduced number of samples available make PLS suitable for appearance-based modeling.

Experimental results demonstrate that the use of an enriched feature set analyzed by PLS reduces the ambiguity among different appearances and provides higher recognition rates when compared to other machine learning techniques. Furthermore, the combination of features usually outperforms the results obtained when individual features are considered.

In addition to appearance-based modeling, we also use the one-against-all scheme for face identification. This problem has received significant attention over the years. For a given probe face, the goal of face identification is to match this unknown face against a gallery of known people. Due to the availability of large amounts of data acquired in a variety of conditions, techniques that are both robust to uncontrolled acquisition conditions and scalable to large gallery sizes, which may need to be incrementally built, are challenges.

In chapter 6 we tackle two problems related to face recognition. Initially, we propose an approach to robust face identification based on PLS to perform multi-channel feature weighting. Then, we extend the method to a tree-based dis-

criminative structure aiming at reducing the time required to evaluate novel probe samples.

The proposed face identification approach outperforms state-of-art techniques in most of the comparisons considering standard face recognition datasets, particularly when the data is acquired under uncontrolled conditions, also supporting that feature combination provides higher discriminative power. Furthermore, the use of PLS is particularly useful in face identification due to the limited number of samples available to describe a subject. We show that our approach provides state-of-art results when only a single sample is available per subject.

Chapter 2

Human Detection Using Partial Least Squares

Two main approaches to human detection have been explored over the last few years. The first class of methods consists of a generative process where detected parts of the human body are combined according to a prior human model. The second class of methods considers purely statistical analysis that combine a set of low-level features within a detection window to classify the window as containing a human or not. The method presented in this chapter belongs to the latter category.

Dalal and Triggs [13] proposed using grids of Histograms of Oriented Gradient (HOG) descriptors for human detection, and obtained good results on multiple datasets. The HOG feature looks at the spatial distribution of edge orientations. However, this may ignore some other useful sources of information, thus leading to a number of false positive detections such as the ones shown in Figure 2.1. Our analysis shows that information such as the homogeneity of human clothing, color, particularly skin color, typical textures of human clothing, and background textures complement the HOG features very well. When combined, this richer set of descriptors helps improve the detection results significantly.

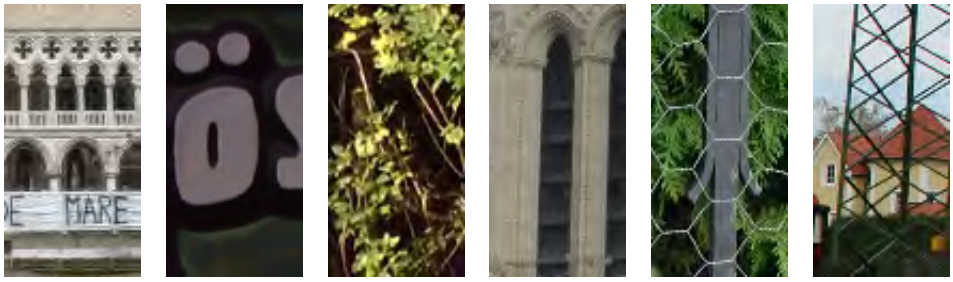


Figure 2.1: False positives obtained when only edge information (using HOG features) is considered.

2.1 Related Work

The work of Dalal and Triggs [13] is notable because it was the first paper to report impressive results on human detection. Their work uses HOG as low-level features, which were shown to outperform features such as wavelets [43], PCA-SIFT [31] and shape contexts [7].

To improve detection speed, Zhu et al. [88] propose a rejection cascade using HOG features. Their method considers blocks of different sizes, and to train the classifier for each stage, a small subset of blocks is selected randomly. Also based on HOG features, Zhang et al. [86] propose a multi-resolution framework to reduce the computational cost. Begard et al. [5] address the problem of real-time pedestrian detection by considering different implementations of the AdaBoost algorithm.

Using low-level features such as intensity, gradient, and spatial location combined by a covariance matrix, Tuzel et al. [70] improve the results obtained by Dalal and Triggs. Since the covariance matrices do not lie in a vector space, the classification is performed using LogitBoost classifiers combined with a rejection cascade

designed to accommodate points lying on a Riemannian manifold. Mu et al. [46] propose a variation of local binary patterns to overcome some drawbacks of HOG, such as lack of color information. Chen and Chen [11] combine intensity-based rectangle features and gradient-based features using a cascaded structure for detecting humans. Applying combination of edgelets [79], HOG descriptors [13], and covariance descriptors [70], Wu and Nevatia [78] describe a cascade-based approach where each weak classifier corresponds to a sub-region within the detection window from which different types of features are extracted. Dollar et al. [14] propose a method to learn classifiers for individual components and combine them into an overall classifier. The work of Maji et al. [40] uses features based on a multi-level version of HOG and histogram intersection kernel SVM based on the spatial pyramid match kernel [33].

Employing part-based detectors, Mikolajczyk et al. [42] divide the human body into several parts and apply a cascade of detectors for each part. Shet and Davis [63] apply logical reasoning to exploit contextual information, augmenting the output of low-level detectors. Based on deformable parts, Felzenszwalb et al. [20] simultaneously learn part and object models and apply them to person detection, among other applications. Tran and Forsyth [69] use an approach that mixes a part-based method and a subwindow-based method into a two stage method. Their approach first estimates a possible configuration of the person inside the detection window, and then extracts features for each part resulting from the estimation. Similarly, Lin and Davis [36] propose a pose-invariant feature extraction method for simultaneous human detection and segmentation, where descriptors are computed adaptively

based on human poses. Mikolajczyk et al. [42] divide the human body into seven parts, and for each part a cascade of detectors is applied.

2.2 Proposed Method

Previous studies [40, 70, 78] have shown that significant improvement in human detection can be achieved using different types (or combinations) of low-level features. A strong set of features provides high discriminatory power, reducing the need for complex classification methods.

Edges, colors and textures capture important cues for discriminating humans from the background. To capture these cues, the low-level features we employ are the original HOG descriptors with additional color information, called *color frequency*, and texture features computed from co-occurrence matrices.

To handle the high dimensionality resulting from the combination of features, PLS is employed as a dimensionality reduction technique. PLS is a powerful technique that provides dimensionality reduction for even hundreds of thousands of variables, accounting for class labels in the process. The latter point is in contrast to traditional dimensionality reduction techniques such as Principal Component Analysis (PCA).

The steps performed in our detection method are the following. For each detection window in the image, features extracted using original HOG, color frequency, and co-occurrence matrices are concatenated and analyzed by the PLS model to reduce dimensionality, resulting in a low dimensional vector. Then, a simple and

efficient classifier is used to classify this vector as either a human or non-human. These steps are explained in the following subsections.

2.2.1 Feature Extraction

We decompose a detection window, d_i, into overlapping blocks and extract a set of features for each block to construct the feature vector $\boldsymbol{v_i}$.

To capture texture, we extract features from co-occurrence matrices [25], a method widely used for texture analysis. Co-occurrence matrices represent second order texture information – i.e., the joint probability distribution of gray-level pairs of neighboring pixels in a block. We use 12 descriptors: angular second-moment, contrast, correlation, variance, inverse difference moment, sum average, sum variance, sum entropy, entropy, difference variance, difference entropy, and directionality [25]. Co-occurrence features are useful in human detection since they provide information regarding homogeneity and directionality of patches. In general, a person wears clothing composed of homogeneous textured regions and there is a significant difference between the regularity of clothing texture and background textures.

Edge information is captured using histograms of oriented gradients. HOG captures edge or gradient structures that are characteristic of local shape [13]. Since the histograms are computed for regions of a given size within the detection window, HOG is robust to some location variability of body parts. HOG is also invariant to rotations smaller than the orientation bin size.

The last type of information captured is color. Although colors may not be consistent due to variability in clothing, certain dominant colors are more often observed in humans, mainly in the face/head regions. In order to incorporate color we used the original HOG to extract a descriptor called *color frequency*. In HOG, the orientation of the gradient for a pixel is chosen from the color band corresponding to the highest gradient magnitude. Some color information is captured by the number of times each color band is chosen. Therefore, we construct a three bin histogram that tabulates the number of times each color band is chosen. In spite of its simplicity, experimental results have shown that color frequency increases detection performance.

Once the feature extraction process is performed for all blocks inside a detection window d_i, features are concatenated creating an extremely high-dimensional feature vector $\boldsymbol{v_i}$. Then, $\boldsymbol{v_i}$ is projected onto a set of weight vectors (discussed in the next section), which results in a low dimensional representation that can be handled by classification methods.

2.2.2 Partial Least Squares for Dimension Reduction

Partial least squares is a method for modeling relations between sets of observed variables by means of latent variables. The basic idea of PLS is to construct new predictor variables, latent variables, as linear combinations of the original variables summarized in a matrix $\boldsymbol{X}$ of descriptor variables (features) and a vector $\boldsymbol{y}$ of response variables (class labels). While additional details regarding PLS meth-

ods can be found in [15, 60], a brief mathematical description of the procedure is provided below.

Let $\mathcal{X} \subset \mathbb{R}^m$ denote an m-dimensional space of feature vectors and similarly let $\mathcal{Y} \subset \mathbb{R}$ be a 1-dimensional space representing the class labels. Let the number of samples be n. PLS decomposes the zero-mean matrix $\boldsymbol{X}$ $(n \times m)$ and zero-mean vector $\boldsymbol{y}$ $(n \times 1)$ into

$$\boldsymbol{X} = \boldsymbol{T}\boldsymbol{P}^T + \boldsymbol{E}$$

$$\boldsymbol{y} = \boldsymbol{U}\boldsymbol{q}^T + \boldsymbol{f}$$

where $\boldsymbol{T}$ and $\boldsymbol{U}$ are $n \times p$ matrices containing p extracted latent vectors, the $(m \times p)$ matrix $\boldsymbol{P}$ and the $(1 \times p)$ vector $\boldsymbol{q}$ represent the loadings and the $n \times m$ matrix $\boldsymbol{E}$ and the $n \times 1$ vector $\boldsymbol{f}$ are the residuals. The PLS method, using the nonlinear iterative partial least squares (NIPALS) algorithm [75], constructs a set of weight vectors (or projection vectors) $W = \{\boldsymbol{w}_1, \boldsymbol{w}_2, \ldots \boldsymbol{w}_p\}$ such that

$$[cov(\boldsymbol{t}_i, \boldsymbol{u}_i)]^2 = \max_{|\boldsymbol{w}_i|=1} [cov(\boldsymbol{X}\boldsymbol{w}_i, \boldsymbol{y})]^2$$

where $\boldsymbol{t}_i$ is the i-th column of matrix $\boldsymbol{T}$, $\boldsymbol{u}_i$ the i-th column of matrix $\boldsymbol{U}$ and $cov(\boldsymbol{t}_i, \boldsymbol{u}_i)$ is the sample covariance between latent vectors $\boldsymbol{t}_i$ and $\boldsymbol{u}_i$. After the extraction of the latent vectors $\boldsymbol{t}_i$ and $\boldsymbol{u}_i$, the matrix $\boldsymbol{X}$ and vector $\boldsymbol{y}$ are deflated by subtracting their rank-one approximations based on $\boldsymbol{t}_i$ and $\boldsymbol{u}_i$. This process is repeated until the desired number of latent vectors had been extracted.

The dimensionality reduction is performed by projecting the feature vector $\boldsymbol{v_i}$, extracted from a detection window d_i, onto the weight vectors $W = \{\boldsymbol{w}_1, \boldsymbol{w}_2, \ldots \boldsymbol{w}_p\}$, obtaining the latent vector $\boldsymbol{z_i}$ $(1 \times p)$ as a result. This vector is used in classification.

The difference between PLS and PCA is that the former creates orthogonal weight vectors by maximizing the covariance between elements in $\boldsymbol{X}$ and $\boldsymbol{y}$. Thus, PLS not only considers the variance of the samples but also considers the class labels. Fisher Discriminant Analysis (FDA) is, in this way, similar to PLS. However, FDA has the limitation that after dimensionality reduction, there are only $c-1$ meaningful latent variables, where c is the number of classes being considered. Additionally, when the number of features exceeds the number of samples, the covariance estimates do not have full rank and the weight vectors cannot be extracted.

2.2.3 Speed Issues

Although detection results can be improved by utilizing overlapping blocks for low-level feature extraction within the detection window, the dimensionality of the feature vector becomes extremely high. As a result, the speed of the human detector decreases significantly due to the time needed to extract features and project them.

To overcome this problem, we employ a two-stage approach. In a fast first stage, based on a small number of features, the majority of detection windows (those with low probability of containing humans) are discarded. The remaining windows are evaluated during a second stage where the complete set of features allows challenging samples to be correctly classified.

The reduced set of features used during the first stage is obtained by selecting representative blocks within the detection window. We use a PLS-based feature selection method called variable importance on projection (VIP) [76] to do this.

VIP provides a score for each feature, so that it is possible to rank the features according to their predictive power in the PLS model (the higher the score the more importance a feature presents). VIP for the j-th feature is defined as

$$\mathrm{VIP}_j = \sqrt{m \sum_{k=1}^{p} b_k^2 w_{jk}^2 / \sum_{k=1}^{p} b_k^2}$$

where m denotes the number of features, w_{jk} is the j-th element of vector $\boldsymbol{w}_k$, and b_k is the regression weight for the k-th latent variable, $b_k = \boldsymbol{u}_k^T \boldsymbol{t}_k$.

The speed improvements are twofold: (i) reducing the overall number of feature computations; (ii) reducing the time to create the data structure for a block, i.e. computing a co-occurrence matrix from which features are extracted. If features were selected individually, then a data structure might need to be constructed for a block to compute only one feature. To avoid that, we select features based on blocks. This way, data structures for a block are only built if several features within the block present some importance.

To obtain the relative discriminative power among blocks we build a PLS model for each block, from which only the first latent variable is considered (since PLS considers class labels, the first latent variable can be used as a clue about how well that block contributes to the detection). A global PLS model is built using as input only the first latent variable of every block. Then, VIP scores are computed with respect to this PLS model, in this way, blocks can be ranked according to their importance in detection. Finally, the features used in the first stage of our approach are those computed from blocks having high rank.

2.3 Experiments

We now present experiments to evaluate several aspects of our proposed approach. First, we demonstrate the need for dimensionality reduction and the advantages of using PLS for this purpose. Second, we evaluate the features used in our system. Third, we compare various classifiers that can be used to classify the data in the low dimensional subspace. Fourth, we discuss the computational cost of our method. Finally, we compare the proposed system to state-of-the-art algorithms on several datasets considering cropped as well as full images.

Experimental Setup. For co-occurrence feature extraction we use block sizes of 16×16 and 32×32 with shifts of 8 and 16 pixels, respectively. We work in the HSV color space. For each color band, we create four co-occurrence matrices, one for each of the (0°, 45°, 90°, and 135°) directions. The displacement considered is 1 pixel and each color band is quantized into 16 bins. 12 descriptors mentioned earlier are then extracted from each co-occurrence matrix. This results in $63,648$ features.

We calculate HOG features similarly to Zhu et al. [88], where blocks with sizes ranging from 12×12 to 64×128 are considered. In our configuration there are $2,748$ blocks. For each block, 36 features are extracted, resulting in a total of $98,928$ features. In addition, we use the same set of blocks to extract features using the color frequency method. This results in three features per block, and the total number of resulting features is $8,244$. Aggregating across all three feature channels, the feature vector describing each detection window contains $170,820$ elements.

We estimate the parameters of our system using a 10-fold cross-validation procedure on the training dataset provided by INRIA Person Dataset [13]. The INRIA person dataset provides a training dataset containing 2416 positive samples of size 64×128 pixels and images containing no humans, used to obtain negative exemplars. We sample this set to obtain our validation set containing 2000 positive samples and 10000 negative samples. In sections 2.3.1 to 2.3.4 our experiments are performed using the INRIA person dataset.

Experimental results using INRIA Person Dataset are presented using detection error tradeoff (DET) curves on log-log scales. The x-axis corresponds to false positives per window (FPPW), defined by *FalsePos/(TrueNeg + FalsePos)* and the y-axis shows the miss rate, defined by *FalseNeg/(FalseNeg + TruePos)*. To clarify the results shown throughout the chapter, curves where the lowest FPPW is 10^{-4} are obtained using the training data, while curves where the lowest FPPW is 10^{-6} are obtained using the testing data.

All experiments were conducted on an Intel Xeon 5160, 3 GHz dual core processor with 8GB of RAM running Linux operating system.

2.3.1 Dimensionality Reduction

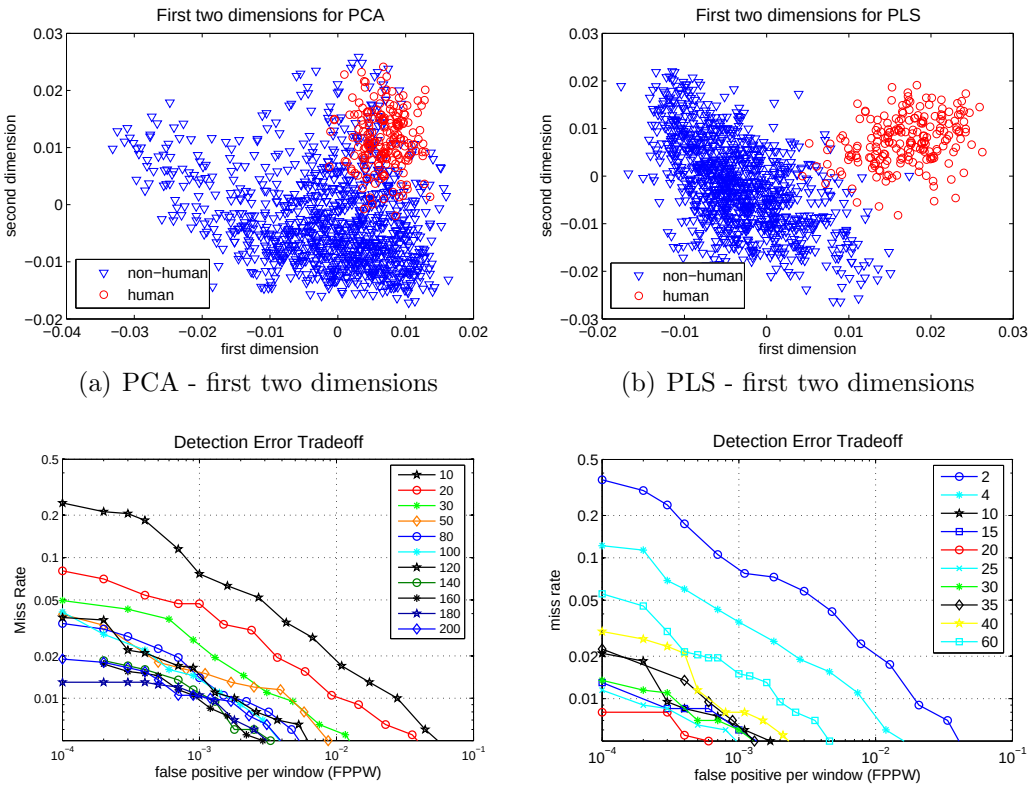
PLS+QDA Vs SVM. We first examine the feasibility of applying support vector machines (SVM) directly on the high dimensional feature space ($170,820$ features per sample). Table 2.1 shows the comparison between time required to train a linear SVM and the time required to train a PLS model along with a Quadratic

Table 2.1: Time, in seconds, to train SVM and PLS + QDA models. The number of features per sample is 170,820. The training time increases with an increase in the number of training samples.

# samples	PLS + QDA	SVM
200	23.63	131.72
600	62.62	733.63
1000	97.38	1693.50
1400	135.81	2947.51
1800	174.57	4254.63
2200	213.93	-
11370	813.03	-

Discriminant Analysis (QDA) model (we use the QDA classifier, but in later subsections we provide a comparison to other classifiers as well). We used the LIBSVM [10] package for this purpose. As the number of training samples is increased, the training time also increases. For more than 1800 samples we were unable to train a linear SVM since the procedure ran out of memory. In addition, the computational cost to learn a PLS model and train a QDA classifier is an order of magnitude smaller than the cost for training an SVM. These results indicate that for such a high dimensional space, it is more suitable to project the data onto a low dimensional subspace and then learn a classifier.

PLS Vs PCA. We now establish a baseline using Principal Component Analysis (PCA) to perform linear dimensionality reduction and compare its results to PLS. Figures 2.2(c) and (d) show the DET curves obtained for a QDA classifier in the PCA subspace as well as in the PLS subspace. It is interesting to note that while the best results are obtained by using the first 20 PLS latent variables, the performance of the system drops when the number of latent variables is increased beyond



(a) PCA - first two dimensions

(b) PLS - first two dimensions

(c) PCA - cross-validation

(d) PLS - cross-validation

Figure 2.2: Comparison of PCA and PLS for dimensionality reduction. (a-b) projection of the first two dimensions of the training samples for one of the models learned in the cross-validation. (c-d) DET curves according to the number of dimensions used to train the classifier.

20. This can be attributed to overfitting of the data caused by using a larger number of latent variables. The results achieved while using the first 20 latent variables are the best results obtained over both subspaces (0.8% miss rate at 10^{-4} FPPW). The best performance on the PCA subspace is obtained for a dimensionality of 180 (1.8% miss rate at 10^{-4} FPPW).

As the dimensionality of the subspace increases, the time required to project the high dimensional feature vectors onto the low dimensional space also increases. On our computer, projecting the feature vector for a single window onto a 180 dimen-

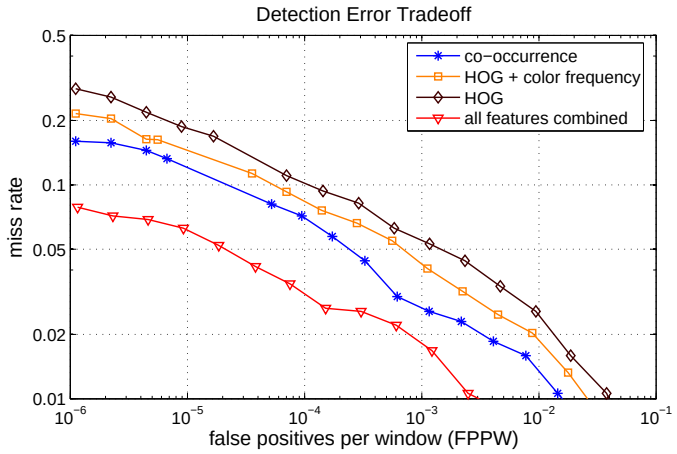


Figure 2.3: Results obtained by using different features and combination of all three feature channels used by this work.

sional PCA subspace takes 0.0264 seconds while it takes 0.0032 seconds to project onto the 20 dimensional PLS subspace. Since an image contains several thousand windows, a computational cost of 0.0264 seconds/window is substantially worse than that for PLS. Thus, in addition to the superior performance, the computational cost of projection makes PLS more suitable for our application than PCA. Figure 2.2(a) and (b) show the training dataset projected onto the first two dimensions for PLS and PCA. PLS clearly achieves better class separation than PCA.

2.3.2 Feature Evaluation

Comparing features. Figure 2.3 shows the results of the three classes of features used in our system as well as the combined performance. We show results combining the HOG and color frequency features to demonstrate the positive contribution of the color features. A significant improvement is achieved when all features

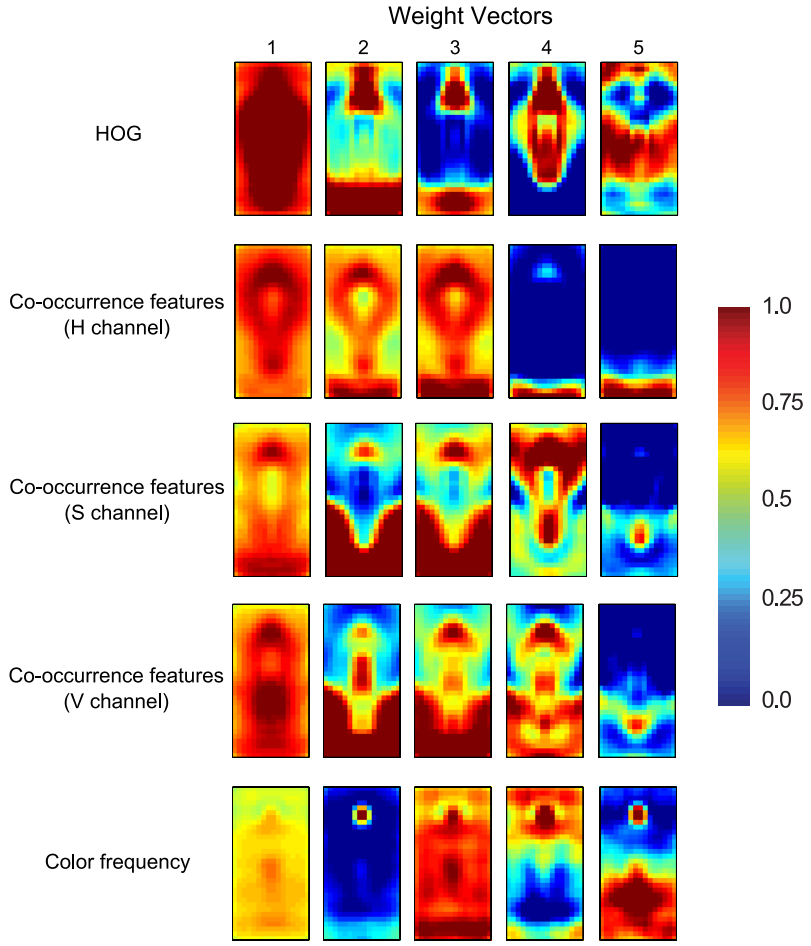


Figure 2.4: Weight vectors for different features within the detection window. Red indicates high importance, blue low (the plots are in the same scale and normalized to interval $[0, 1]$).

are combined.

Analysis of the PLS Weight Vectors. In this experiment, we perform an analysis of the contribution of each feature channel based on the weights of the PLS weight vectors used to project the features onto the low dimensional subspace. We use the same idea as described in Section 2.2.3. For a given block in the detection window, we create a PLS model for each feature channel. Then, using only the first latent variable for every block, we learn a global PLS model. Figure 2.4 shows the

weights for the first five projection vectors of this global PLS model. The features considered are HOG, co-occurrence extracted from color bands H, S and V, and the color frequency.

Figure 2.4 shows how each feature channel (edge, texture, color) provides information from different regions within the detection window. This supports our claim that the considered features complement each other, leading to an improvement over single-feature-based methods. For example, the first weight vector of the HOG feature set captures information about the body shape due to the presence of edges. Co-occurrence matrix features from color band H extract information around the body silhouette. Color bands S and V provide information about the head and homogeneous parts inside the body, respectively. Except for the first weight vector, color frequency features are able to identify regions located in the head due to similarity of the dominant colors in that region (skin color).

2.3.3 Classification in Low Dimensional Space

To evaluate the classification in the low dimensional subspace, we compare the performance of several classifiers using the 10-fold cross-validation described earlier. Figure 2.5 shows the results. According to the results, QDA classifier, kernel SVM and linear SVM achieved comparable performance in low dimensional subspace. Due to its simplicity, we have chosen to use QDA in our system. PLS tends to produce weight vectors that provide a good separation of the two classes for the human detection problem, as shown in Figure 2.2(b). This enables us to use simple

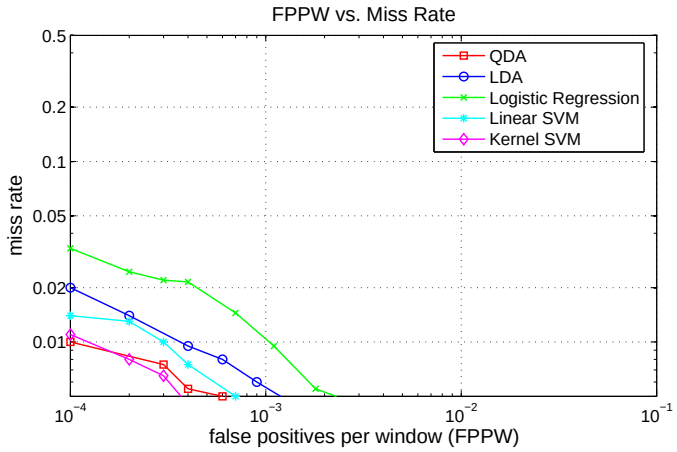


Figure 2.5: Comparison of several classification methods for the low dimensional PLS subspace.

classifiers in the low dimensional subspace.

2.3.4 Computational Cost

We accelerate the process using the two-stage approach described in Section 2.2.3. To reduce the number of features computed in the first stage, we rank blocks according to their VIP scores and then select only those features in blocks with higher rankings. Using 10-fold cross-validation in the training set, we select a subset of blocks containing $3,573$ features per detection window, together with a probability threshold to decide whether a detection window needs to be considered for the second stage.

It is important to note that the use of the first stage alone achieves poor results for low false alarm rates. Therefore, for the detection windows not discarded in the first stage (approximately 3% for the INRIA person dataset), the complete feature

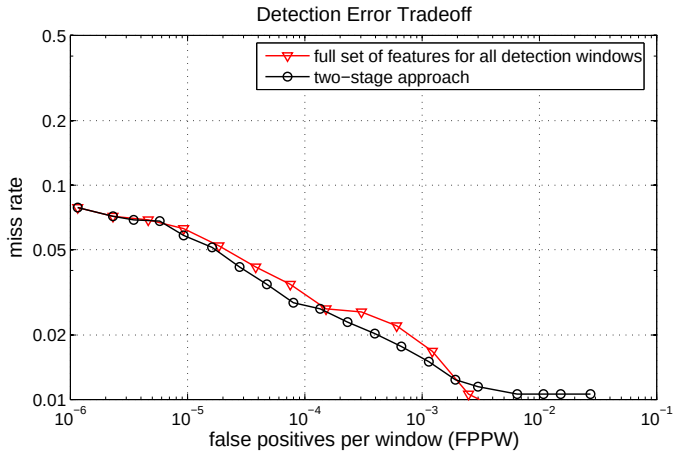


Figure 2.6: Results after adding two stages compared to results obtained without speed optimization.

set is computed. For the testing set of the INRIA person dataset, the results shown in Figure 2.6 indicate no degradation in performance at low false alarm rates when the two-stage approach is used, as compared to computing the full set of features for all detection windows. After speeding the process up using our two-stage method, we were able to process 2929 detection windows per second.

2.3.5 Evaluation and Comparisons

In this section we evaluate the proposed system on different datasets and compare it to state-of-the-art methods.

INRIA Person Dataset. The INRIA person dataset [13] provides both training and testing sets containing positive samples of size 64×128 pixels and negatives images (containing no humans). To estimate weight vectors (PLS model) and train the quadratic classifier we employ the following procedure. First, all

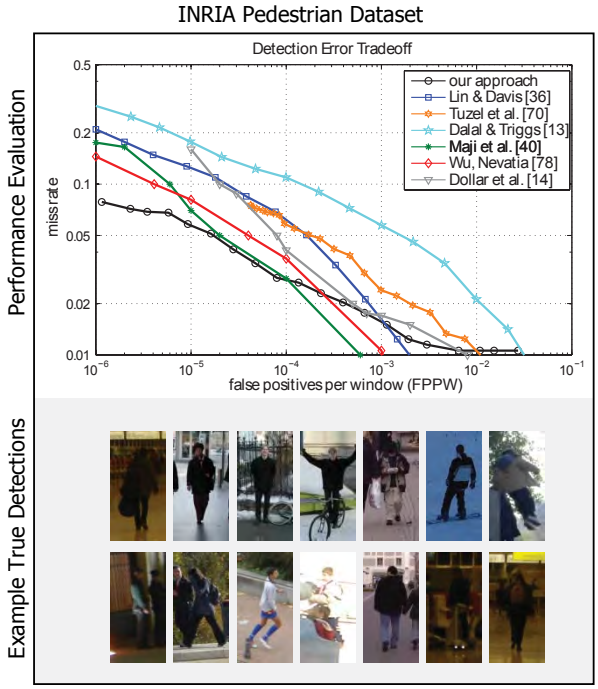


Figure 2.7: Evaluation of our method using the INRIA Pedestrian Dataset. First row shows performance and comparisons with state-of-the-art methods. Second row shows some sample true detections for the dataset.

2416 positive training samples and 5000 of the negative detection windows, sampled randomly from training images, are used. Once the first model is created, we use it to classify negative windows in the training set. The misclassified windows are added into the 5000 negative windows and a new PLS model and new classifier parameters are estimated. This process is repeated a few times and takes approximately one hour. Our final PLS model considers 8954 negative and 2416 positive samples, using 20 weight vectors (as discussed in section 2.3.1).

Figure 2.7 compares results obtained by the proposed approach to methods published previously. Our results were obtained using 1126 positive testing samples

and by shifting the detection windows by 8 pixels in the negative testing images, all of which are available in the dataset. While we were able to run the implementations for methods [13, 70], curves for methods [14, 36, 40, 78] were obtained from their reported results. The PLS approach outperforms all methods in regions of low false alarm rates, i.e. 5.8% miss rate at 10^{-5} FPPW and 7.9% miss rate at 10^{-6} FPPW.

DaimlerChrysler Pedestrian Dataset. This dataset provides grayscale samples of size 18×36 pixels [47]. We adapt our feature extraction methods for these image characteristics as follows. For co-occurrence feature extraction, we use block sizes of 8×8 and 16×16 with shifts of 2 pixels for both. Co-occurrence matrices are estimated using the brightness channel quantized into 16 bins. For HOG feature extraction, we adopt the same approach used for the INRIA person dataset; however, block sizes now range from 8×8 to 18×36. Due to the lack of color information, the color frequency feature cannot be considered in this experiment.

The DaimlerChrysler dataset is composed of five disjoint sets, three for training and two for testing. To obtain results that can be compared to those presented by Maji et al. [40] and by Munder and Gavrila [47], we report results by training on two out of three training sets at a time. Therefore, we obtain six curves from which the confidence interval of the true mean detection rate is given by the $t_{(\alpha/2,N-1)}$ distribution with desired confidence of $1 - \alpha = 0.95$ and $N = 6$. The boundaries of this interval are approximated by $\overline{y} \pm 1.05s$, where $\overline{y}$ and s denote the estimated mean and standard deviation, respectively [47].

Figure 2.8 compares results obtained by the proposed method to results reported in [40, 47]. In contrast to previous graphs, this shows detection rates instead

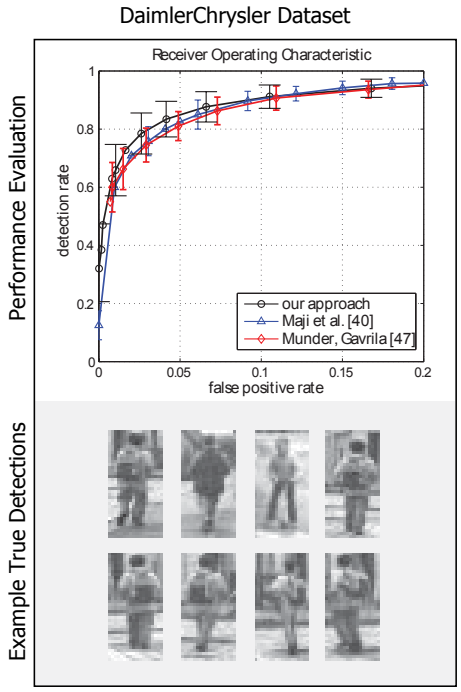


Figure 2.8: Evaluation of our method on the DaimlerChrysler Dataset. First row shows performance and comparisons with state-of-the-art methods. Second row shows some sample true detections for the dataset.

of miss rates on the y-axis and both axes are shown using linear scales. Similar to experiments conducted on the INRIA person dataset, the results obtained with the proposed method show improvements in regions of low false alarm rates.

ETHZ Dataset. We evaluate our method for un-cropped full images using the ETHZ dataset [17]. This dataset provides four video sequences, one for training and three for testing (640×480 pixels at 15 frames/second). Even though a training sequence is provided, we do not to use it; instead we use the same PLS model and QDA parameters learned on the INRIA training dataset. This allows us to evaluate the generalization capability of our method to different datasets.

For this dataset we use false positives per image (FPPI) as the evaluation

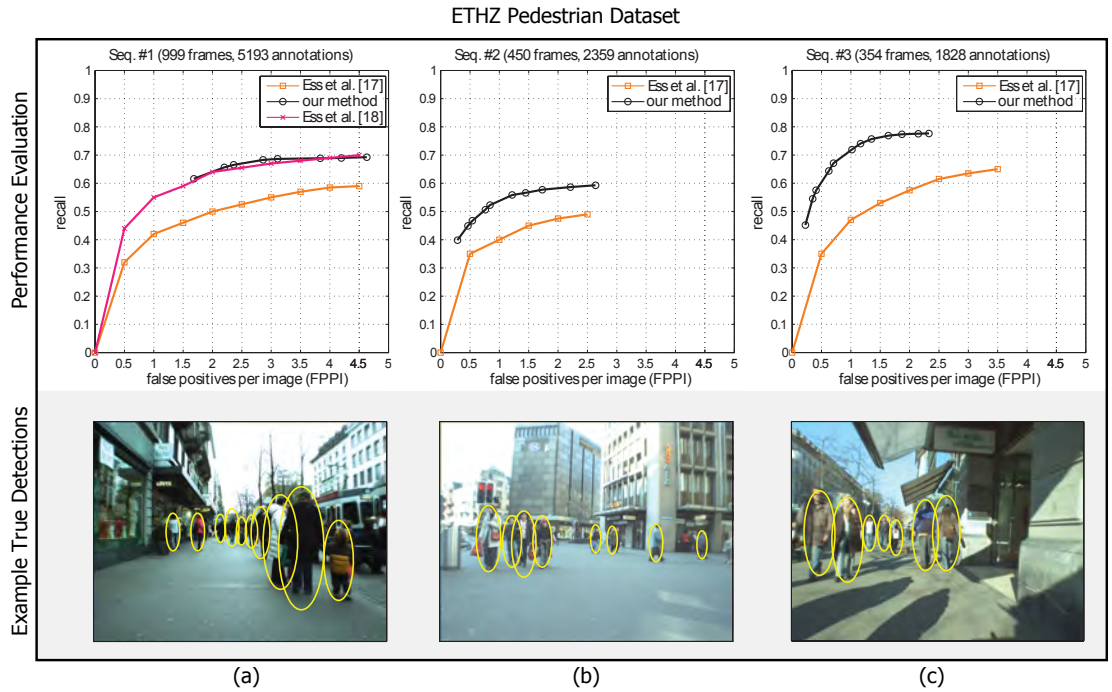
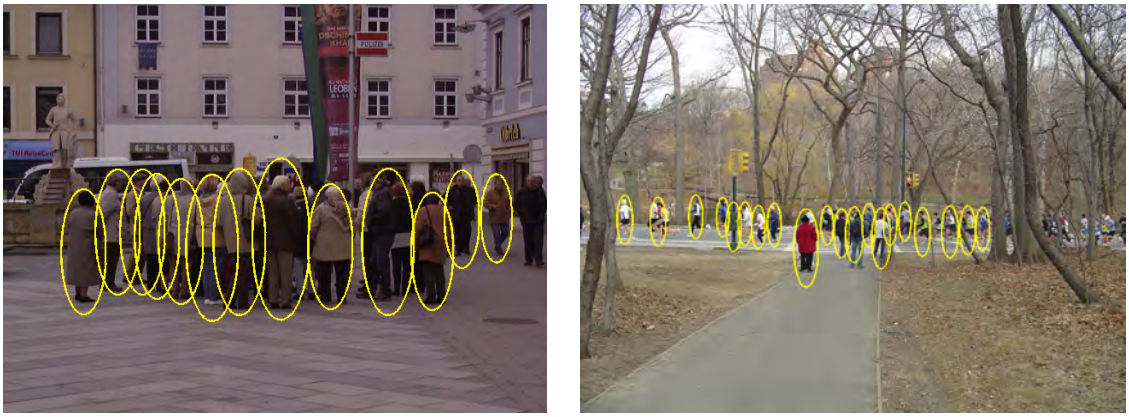


Figure 2.9: Evaluation of our method on the ETHZ Pedestrian Dataset. First row shows performance and comparisons with state-of-the-art methods. Second row shows some sample true detections for the dataset.

metric, which is more suitable for evaluating the performance on full images [69]. Using the same evaluation procedure described in [17] we obtain the results shown in Figure 2.9 for the testing sequences provided. We use only the images provided by the left camera and perform the detection for each single image at 11 scales without considering any temporal smoothing. We do not train our detector on the provided training set and we do not use any additional cues such as depth maps, ground-plane estimation, and occlusion reasoning, all of which are used by [17]. Yet, our detector outperforms the results achieved by [17] in all three video sequences.

The work by Ess et al. [18] also considers sequence #1 in their experiments, so we have added their results in Figure 2.9(a). Even though [18] uses additional cues



(a) 640 × 480 (41,528 det. windows)

(b) 1632 × 1224 (389,350 det. windows)

(c) 1600 × 1200 (373,725 det. windows)

(d) 800 × 533, (61,820 det. windows)

Figure 2.10: Results obtained from images containing people of different sizes and backgrounds rich in edge information. The image size and the total number of detection windows considered are indicated in the caption.

such as tracking information, our method, trained using the training set of INRIA dataset, achieves very similar detection results.

Additional Set of Images. We present some results in Figure 2.10 for a few images obtained from INRIA testing dataset and Google. These results were also obtained using the same PLS model and QDA parameters learned on the INRIA training dataset. We scan each image at 10 scales. Despite the large number of detection windows considered, the number of false alarms produced is very low.

Chapter 3

Human Detection Under Occlusion by Integrating Face and Person Detectors

As mentioned earlier, human detection in still images is a challenging problem due to the presence of variations in people's poses, lighting conditions, inter- and intra- person occlusion, amongst others. Occlusion, in particular, poses a significant challenge due to the large amount of variations it implies on the appearance of the visible parts of a person.

Subwindow-based person detectors present degraded performance when parts of the body are occluded; part-based approaches, on the other hand, are better suited to handle such situations because they still detect the un-occluded parts. However, since part-based detectors are less specific than whole body detectors, they are less reliable and usually generate large numbers of false positives. Therefore, to obtain more accurate results it is important to aggregate information obtained from different sources with a part-based detector. For this, we incorporate a face detector.

Face detection is an extensively studied problem, and the survey paper [80] provides a comprehensive description of various approaches to this problem. For example, Viola and Jones [72] use large training exemplar databases of faces and non-faces, extract feature representations from them, and then use boosting techniques

to classify regions as face or non-face. Other algorithms, for instance Rowley et al. [61], uses a neural network to learn how the appearance of faces differ from non-faces using training exemplars, and then detect faces by seeing how well the test data fits the learned model. Another class of approaches, exemplified by Heisele et al. [26], uses a part-based framework by looking for prominent facial components (eyes, nose etc), and then uses their spatial relationship to detect faces. Although such methods are more robust to image deformations and occlusions when compared with holistic approaches, the choice of feature representations and accurate characterization of the relationships between the facial components is still a challenge.

The question that arises naturally is then, how to fuse these two sources to improve overall detection performance. Specifically, is it possible to use the response profiles of the two separate detectors, to reinforce each other, as well as provide a basis to resolve conflicts? This is the question we address here. Figure 3.1 motivates the utility of combining face and person detectors. First, while the lower half of person c is occluded, the face detector can still detect the face of the person, whereas the person detector might fail. Nevertheless, we can try to *explain* the response of the person detector based on the response of the face detector, and conclude that a person is present. Another case is the reverse situation such as b and d in Figure 3.1 whose faces are partially occluded while the body parts are completely visible. Such situations occur often in real-world scenarios, and motivates exploring *feedback* between face and people detectors.

Figure 3.1: Image where occlusion is present and fusion of detectors can increase detection accuracy. Face of person b is occluded. Once the legs and torso are visible, results from a part-based person detector can be used to support that a human is present at that location. On the other hand, the legs of person c are occluded, in such a case, face detector results can be used to reason that there is a person at that particular location since the face of person c is perfectly visible.

3.1 Face and Person Detection

In this section we give a synopsis of our algorithms for face detection and person detection. We also provide detection results of applying the individual algorithms on standard datasets, showing that these detectors individually achieve results comparable to state-of-art methods. However, a point to keep in mind is that these standardized datasets do not have considerable amounts of occlusion, which is the main problem that we address here.

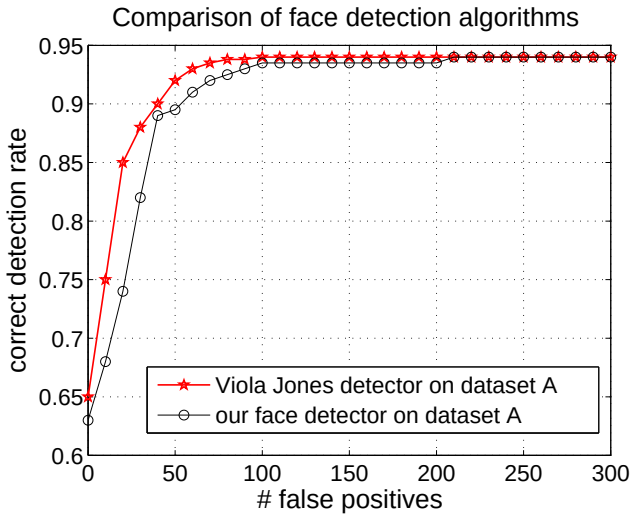


Figure 3.2: Experimental results for face detection on the MIT+CMU dataset.

3.1.1 Face Detection

We use a feature-based approach to detect faces from still images. Our approach, motivated by [44], is based on using an optimal step edge operator to detect shapes (here, the facial contours are modeled as ellipses). The crux of the algorithm is then to obtain the edge map of the image using a derivative of double exponential (DODE) operator, and fit various sized ellipses to the edge map. Image regions that have high response to ellipse fitting signify locations that likely contain faces.

We then conduct post-processing on these short-listed regions by computing three different cues – color [28], histogram of oriented gradients [13], and eigenfaces [6], and combine the three feature channels using support vector machines [50] to decide whether a face is present or not. The motivation behind the choice of these descriptors is: (i) the human face has a distinct color pattern which can be characterized by fitting Gaussian models for the color pattern of face regions, and

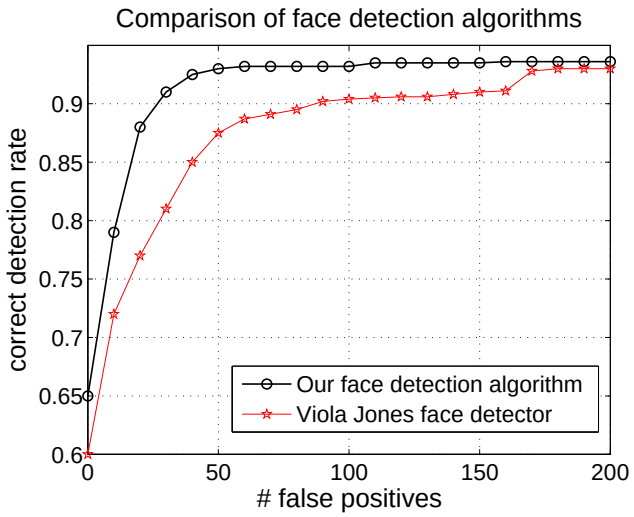


Figure 3.3: Experimental results for face detection on the maritime dataset.

non-face regions; (ii) the histogram of oriented gradients capture the high interest areas in faces that are rich in gradient information (eyes, nose and mouth) that are quite robust to pose variations, and (iii) eigenfaces captures the holistic appearance of the human face. These three feature channels capture a mix of global and local information about the face, and are robust to variations in pose.

Our algorithm was tested on the MIT+CMU face dataset [61]. This dataset has two parts. The first part (A) has 130 frontal face images with 507 labeled faces, the second part (B) has 208 images containing 441 faces of both frontal and profile views. The results of our algorithm are presented in Figure 3.2. Most other algorithms that are evaluated on this dataset do not provide the full ROC, but rather provide certain points on the ROC. Since Viola and Jones [72] quote their ROC for part A of this dataset, we have compared our ROC with theirs; even otherwise, it can be observed that our performance is comparable to the ROC points of other

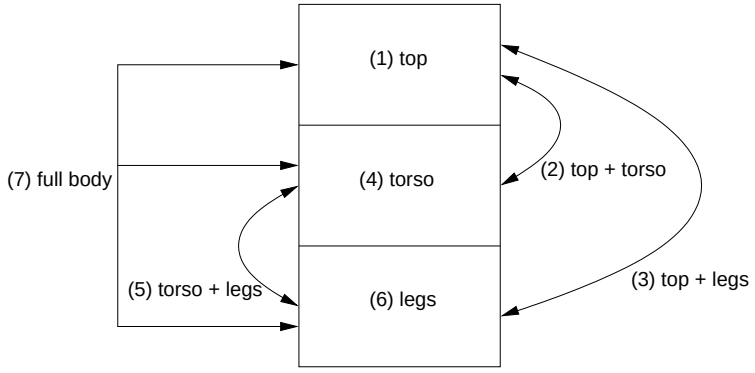


Figure 3.4: Parts of a detection window used to train multiple detectors.

algorithms (like Rowley et al. [61]). Since we are interested in detecting partially occluded faces we also compare our approach to the OpenCV implementation of Viola and Jones [72] method on the internally collected maritime dataset in Figure 3.3.

3.1.2 Person Detection

For person detection we use the PLS detector described in Chapter 2. Since part-based approaches are better suited to handle situations of occlusion, we split the person detector into seven different detectors, which consider the following combinations of regions of the body: (1) top, (2) top-torso, (3) top-legs, (4) torso, (5) torso-legs, (6) legs, and (7) full body, as illustrated in Figure 3.4. Therefore, at each position in the image the person detector estimates a set of seven probabilities. The training for these detectors was performed using the training set of the INRIA person dataset.

As discussed earlier, part-based approaches for person detection have been employed previously. Here, we use a part-based approach in tandem with a face

detector creating a small number of intuitive case-based models for overall person detection.

Although the face and person detectors present results comparable to the state of the art on these datasets, these algorithms face difficulties when there is significant occlusion. To this end, we explore how to overcome this problem by combining the responses of the individual detectors.

3.2 Integrating Face and Person Detection

In this section we present our algorithm for integrating the response profiles of face and person detectors. We model observations of the individual detectors, and generate hypotheses that capture intuitive relationships between the responses of the face detector and the person detector. Specifically, we describe a set of situations where the output of one detector can be logically combined with the other detector's output to eliminate false alarms or confirm true positives.

3.2.1 Modeling the Response Profiles of the Individual Detectors

To integrate person and face detectors' output we first create models according to the probability profile resulting from individual detectors (the seven probabilities from part-composition person detector and one from the face detector).

For the person detector, we summarize the probability profile obtained by the seven probabilities into a set of four models that inherently capture situations in which various combinations of face and person parts are detected with *high* proba-

bility. Specifically,

Model M_1: all body parts are visible

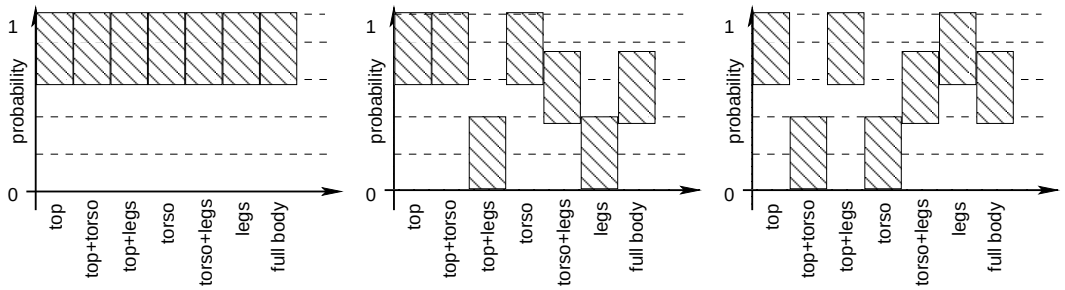
Model M_2: top is visible, torso and legs may or may not be visible. This corresponds to the typical situation in which a person's legs are occluded by some fixed structure like a desk, or the railing of a ship.

Model M_3: top is invisible, whereas torso and legs are visible

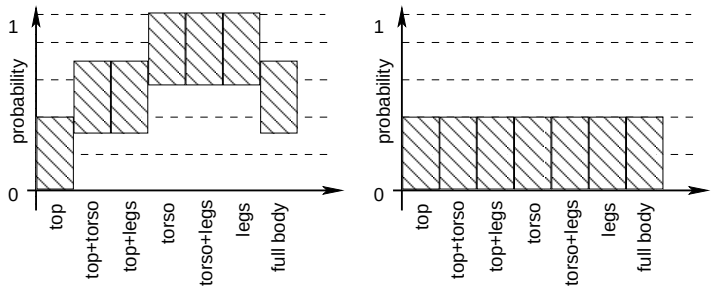
Model M_4: all body parts are invisible

Given the set of seven probabilities estimated by the person part-combination detectors, we define probability intervals that characterize each model. The estimation of the intervals for models M_1 and M_4 can be done automatically by evaluating probability of training samples from standard person datasets. However, probability intervals for models M_2 and M_3 only can be estimated if a training set containing partially occluded people were available. Due to the absence of such dataset, we define the probability intervals for M_2 and M_3 manually.

Figure 3.5 shows the probability intervals for each model. A model M_i fits a detection window if all seven estimated probabilities fall inside the probability intervals defined by M_i. We also estimate a degree of fit of a detection window to each model by simply counting the number of probability intervals satisfied by the response profile:



(a) M_1: all parts are visible (b) M_2: top part is visible and torso is visible (c) M_2: top part is visible and legs are visible



(d) M_3: top part is invisible (e) M_4: all parts are invisible

Figure 3.5: Models designed considering the output profile of the person detector. The x-axis has the seven detectors and the y-axis the probability interval for each one according to the model. Note that M_2 has two sub-cases, shown in (b) and (c).

$$f(M_i) = \sum_{j=1}^{7} \begin{cases} 1 & \text{if } u_{i,j} \leq P_j \leq l_{i,j} \\ 0 & \text{otherwise} \end{cases} \tag{3.1}$$

where P_j denotes the probability estimated by the j-th detector, $u_{i,j}$ is the upper bound for the j-th interval defined for M_i and $l_{i,j}$ denotes the lower bound. Therefore, we can rank the models according to how well they fit a given detection window. We say that a model M_i has a rank higher than M_j when $f(M_i) > f(M_j)$.

For the face detector, the observations are characterized by the probability values indicating the presence of face for a given detection window. According

to this probability we define three models. We say that a face is present if the probability exceeds a certain threshold (model F_1). We also consider the case when the probability is smaller than the threshold but not negligible (i.e. face might be partially occluded), we refer to this as model F_2. Model F_2 is interesting when the person detector gives a response that supports the low (but not negligible) confidence of the face detector. Finally, we say that a sample fits model F_3 if the probability of face detector is very low.

3.2.2 Generating Hypotheses to Integrate Detectors

Now that we have designed models according to the response profiles to capture occlusion situations, we create a set of hypotheses (rules) to characterize the relation between the detector responses so that these different sources of information can be used to *verify* each other's output. We separate the possibilities into five different hypotheses. The first two hypotheses describe the scenario where the person detector (PD) is used to *verify* the output of the face detector (FD), and the remaining three hypotheses deal with the alternate scenario of using face detector to verify the person detector outputs. The hypotheses are described in the form of conditional rules as follows.

$H_1 : [(f(M_1) \wedge f(M_2)) > (f(M_3) \wedge f(M_4))|F_1]$ Given that the face detector provides high response for a detection window, we look at the models that characterize the person detector output. Since the face is visible, the output of PD should better fit models M_1 or M_2 than M_3 and M_4 since we expect the top (head

and shoulder) features to be detected by the person detector. If that is the case, then PD output verifies that the FD output is correct. Thus, a person is present at that location.

$H_2 : [(f(M_3) \vee f(M_4)) > (f(M_1) \wedge f(M_2))|F_1]$ The alternate case is, given high response for the face detector, if the output of PD fits either M_3 or M_4, then PD indicates that the face is not visible, and hence the output of the FD is a false alarm.

$H_3 : [(F_1|(f(M_1) \vee f(M_2)) > (f(M_3) \wedge f(M_4))]$ Given that the rank of M_1 or M_2 is greater than M_3, if FD gives a high response, then the face detector is reinforcing the output of the person detector. Thus, we conclude that a person is present at the corresponding location.

$H_4 : [(F_2|f(M_3) > (f(M_1) \wedge f(M_2) \wedge f(M_4))]$ A slightly different case from H_3 is when FD has low response, but still has some probability higher than 0 but not high enough to conclude the presence of face. In this case, if for the person detector the rank of M_3 is higher than M_1, M_2, and M_4, then we still decide that there is a person whose face is partially occluded. This is because M_3 captures the situation where the face is occluded, while the torso and legs are visible.

$H_5 : [F_3|(f(M_1) \vee f(M_2) \vee f(M_3)) > f(M_4)]$*:* This final hypothesis deals with the case where the output of person detector fits either M_1, M_2, or M_3, and the probability outputted by the face detector is negligible, so that it cannot come

under H4. In such a case, since the face is completely invisible, we decide that the PD output is a false alarm.

Essentially, the above hypotheses are built on the fact that the presence of the face implies the presence of a person and vice-versa. We do need some confidence value for the presence of face to make decisions on the output of the person detector. This is based on our observation that the presence of just the torso and legs with no information regarding the face is not a strong cue to detect a person. This condition gives rise to many false alarms.

3.3 Experimental Results

In this section, we demonstrate with experiments how our integration framework improves detection under occlusion, as well as reduces the false alarms. We tested our algorithm on challenging images taken from an internally collected maritime dataset. It contains images of 3008×2000 pixels, which is suitable for face and person detection, unlike standard datasets used for person detection, which in general contain images with resolution too low to detect faces. This dataset is a good test-bed since it provides challenging conditions wherein the individual face/person detector might fail, thereby emphasizing the need to fuse information obtained by these detectors.

We now present several situations where the integration framework helps to detect humans. In the image shown in Figure 3.6(b) a person detector would fail to detect people seated since the lower body is occluded. However, our framework

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

Figure 3.6: Results on images from maritime dataset.

combines face information with the presence of the top part of the body (head and shoulders) captured by the person detector. Therefore, it concludes that a person is present. Additionally, Figures 3.6(c), (e), and (f) contain people who are partially occluded. Such conditions would reduce significantly the probability estimated by

an independent person detector, whereas the integration helps resolve this problem.

Next, if the face is partially occluded, then the person detector output will belong to model M_3, whereas face detector's output will have some small value that is not very high and not negligible either. In this case, the person detector results can be used to identify the presence of the face. For example, Figures 3.6(d) and (f) contain people whose faces are occluded. In these cases a face detector would fail to give a high response, but the proposed framework overcomes this problem by aggregating information from body parts.

Essentially, since we are using two separate detectors, if the observations of the person detection and face detection provide conflicting information, then our framework mitigates false positives. A typical example is when hypothesis H_2 is satisfied, which can be used to correct the false alarm of the face detector, and when hypothesis H_5 is satisfied, that helps in reducing the false alarms of the person detector. Additionally, if both individual detectors denote the presence of a person, detection is more reliable than when relying on only one detector.

We tested our algorithm on 20 images containing 126 people. Figure 3.7 presents the detection error tradeoff of our integration method and compares its results to individual detectors. It can be seen that the use of the proposed method results in a substantial improvement in detection accuracy/false alarm suppression. To generate the curve for the our algorithm, we fix the threshold for the face detector and for the person detector we measure how well each model fits a sample by

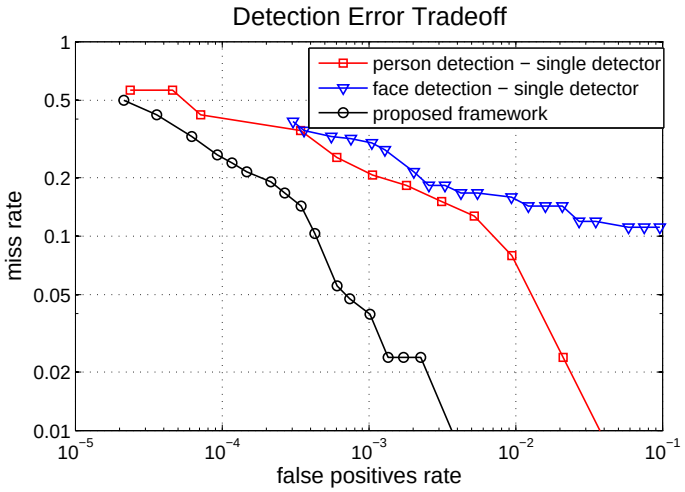


Figure 3.7: Detection error tradeoff comparing the integration to individual face and person detectors. The proposed framework outperforms the individual detectors for all points on the curve.

$$g(M_i) = \frac{1}{7}\sum_{j=1}^{7}\begin{cases} |P_j - u_{i,j}| & \text{if } P_j > u_{i,j} \\ |P_j - l_{i,j}| & \text{if } P_j < l_{i,j} \\ 0, & \text{otherwise} \end{cases} . \tag{3.2}$$

With this equation we obtain values of $g(M_i)$ for every sample. Then, varying a threshold value from zero to one we are able to evaluate which hypotheses are satisfied at each step.

Chapter 4

A Data-Driven Detection Optimization Framework

High accuracy and low computational cost are desirable properties for object detection systems. The need for low computational cost is dictated by the large amount of data that needs to be processed, i.e. object detection may be applied on thousands of frames of a surveillance video. In general, there is a trade-off between accuracy and computational cost, in which the achievement of higher detection speeds results in some degradation of accuracy.

Object detectors are learned using datasets representing different types of scenes – e.g., indoor, outdoor, and urban – to be as general as possible during the detection phase, so that they can detect objects in scenes with very different characteristics. To accomplish this, detection methods usually combine different types of strong feature descriptors, which leads to a higher computational costs. On the other hand, simpler and fewer features can provide enough information to perform accurate detection when scene-specific characteristics are considered during the detector's learning, i.e. a detector is learned to be used specifically with a video feed from a fixed camera pointing towards a parking lot. As a result, the computational cost of the scene-specific detector would be lower, but it would be unlikely to work in scenes with different characteristics due to the bias incorporated in the training set. Therefore, it is worth seeking a balance between learning done

completely offline (and general enough to be used in multiple environments), and learning considering specific characteristics of a scene, so that accurate detection results can be obtained at lower computational cost.

Here we integrate a multi-stage detector learned using very general training datasets (referred as *generic detector*), with regression models learned based on information extracted from the video sequence of a specific scenario. The idea is to use the response of the generic detector, considered to be accurate due to its use of strong feature descriptors, to learn and update data-driven regression models. These regression models are based on efficiently computable features and is used to estimate the generic detector's output (both the response of the detector and the precise location of objects). This will allow us to reject large number of detection windows without extracting expensive feature descriptors used by the generic detector.

Specifically, the regression models use feature descriptors as independent variables and two types of dependent variables. The first type estimates responses of the generic detector at each stage using efficiently computable features in order to reject detection windows quickly. The second type estimates the location of objects. According to our experiments, at each stage multiple detection windows located near the correct location of objects are selected for the next stage. The regression based on the second type of dependent variables is used to obtain a better estimate of the correct location of objects, so that redundant detection windows can be rejected, reducing consequently the number of detection windows evaluated by the generic detector, which is applied to remaining detection windows.

We use the PLS human detector described in chapter 2 as the generic detector.

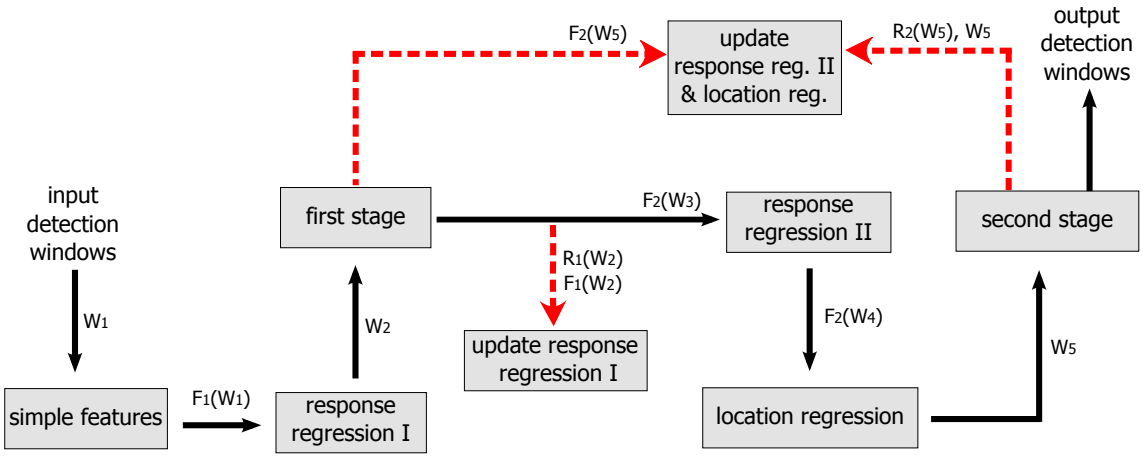


Figure 4.1: Integration of the regression models into the PLS detector. Regression models to estimate the detector's response, *the response regression I* and *the response regression II*, are added before each stage and a regression to estimate the object location, *the location regression*, is incorporated right before the second stage.

The PLS detector provides high accuracy across different datasets but with still high computational cost. Its accuracy is due to the use of several feature channels combined by partial least squares (PLS). The online data-driven regression models proposed here also use PLS, but PLS regression instead of PLS as a dimensionality reduction tool as in the generic detector. Three regression models are considered: two to estimate the generic detector's responses before each stage and one to estimate the human's location, performed right before the second stage of the PLS detector. The flowchart of the integrated method is shown in Figure 4.1. Its details are described later.

4.1 Related Work

A common approach used to optimize object detection is based on a boosted cascade composed of weak classifiers learned during the training phase. In their seminal work,Viola and Jones [71] propose a face detector based on AdaBoost [21], that combines successively more complex classifiers in the cascade so that a large number of detection windows can be excluded in earlier (and faster) stages of the cascade. More recently, Zhu et al. [88] extracted histogram of oriented gradients features using a cascade classifier framework obtaining real-time detection with detection accuracy comparable to the HOG detector proposed by Dalal and Triggs [13] at the expense of largely increased training time. In addition, based on the covariance features proposed by Tuzel et al. [70], Paisitkriangkrai et al. [52] propose a cascade classifier that considers weak classifiers in the Euclidean space instead of on the Riemannian manifold, which provides faster computation.

Looking at the data in multiple resolutions has also been an approach to optimize object detection. Using a predefined feature hierarchy, where lower resolution features are initially used to reject the majority of negative windows at relatively low cost leaving a small number of detection windows to be processed in higher resolutions, Zhang et al. [86] consider HOG features for object detection. As a result, they achieve real-time detection with performance comparable to the HOG detector. Not only using feature hierarchy but also considering the spatial stride of the sliding window search inversely proportional to the features resolution, Pedersoli et al. [53] propose a detection method that provides even higher speed-ups since fewer

detection windows need to be considered.

Another approach to optimize the detection is to develop or change feature descriptors so that they present lower computational cost. Pettersson et al. [54] propose the HistFeat to reduce the memory bandwidth for evaluating the HOG features. Only one memory access is required to perform a feature evaluation, differently from six to nine accesses when the features are evaluated from an integral image. The HistFeat was extended later by Overett et al. [51] aiming at a better balance in terms of processing and memory bandwidth. Abramson et al. [1] propose the control-points features, based on relations between sets of pixels avoiding the need to compute histograms over regions of the image. These features are extended and called connected-control-points in [45], by adding constrains on the location of the points selected to be part of the relations.

Due to the availability of parallel processing units, several works seek for optimization using features provided by the parallel architecture provided by graphics processing units (GPU). Wojek et al. [74], Zhang and Nevatia [84], and Prisacariu and Reid [58] implement in GPU versions of the HOG detector. All three works obtain detection accuracy similar to the CPU implementation but significant speed-ups, i.e. speed-up of 67× are reported by Prisacariu and Reid [58].

The main difference between the approach proposed in this work and the approaches described is that our method performs online learning, incorporated by using the regression models. Therefore, differently from these approaches, the characteristics of the video in which the detection is being performed are explicitly used by our method to achieve lower computational cost.

Several works also consider online learning [29, 32, 35, 77]. In general, they consider an initial (possible weakly) trained model and update its parameters as the detection is being performed. These works usually focus on improvements on accuracy rather than computational cost reduction. Even though we are not updating the generic detector and our goal is not improvement on detection accuracy, experiments show that the incorporation of regression models not only provides lower computational cost but also higher detection rates.

4.2 Proposed Method

In this section we first present a brief review of partial least squares regression and an brief overview of the PLS detector discussing particularly its inputs and outputs for each stage, which will be used to learn the regression models. Then, the data-driven optimization framework based on regression models and its integration with the PLS detector are described.

4.2.1 Partial Least Squares Regression

Once the low dimensional representation of the data has been obtained by NIPALS, as described in Section 2.2.2, the regression coefficients $\boldsymbol{\beta}_{m\times 1}$ can estimated by

$$\boldsymbol{\beta} = \boldsymbol{W}(\boldsymbol{P}^T\boldsymbol{W})^{-1}\boldsymbol{T}^T\boldsymbol{y}. \tag{4.1}$$

The regression response, y_v, for a feature vector $\boldsymbol{v}$ is obtained by

$$y_v = \overline{y} + \boldsymbol{\beta}^T\boldsymbol{v} \tag{4.2}$$

where $\overline{y}$ is the sample mean of $\boldsymbol{y}$.

It is important to point out that even though the number of latent vectors used to create the low dimensional representation of the data matrix $\boldsymbol{X}$ is p (possibly greater than 1), Equation 4.2 shows that only a single dot product of a feature vector with the regression coefficients is needed to obtain the response of a PLS regression model. This makes the use of PLS regression particularly fast to obtain a estimation of the detector's response for a detection window when compared to the original detector, which needs to perform several dot products before obtaining the response.

4.2.2 Overview of the PLS Detector

The first stage of the PLS detector (referred in this chapter as *first stage*) considers a small subset of the 170,820 features in order to reject detection windows faster. Then, for the detection windows not rejected by the first stage (detection windows with response higher than a predefined threshold), the full feature set is extracted and a second response value is obtained (this stage will be referred as *second stage*). Finally, detection windows with high responses are identified by the second stage as the location of the humans in the image.

Figure 4.2 shows a diagram illustrating the flow of the PLS detector. It is important to note that although the first stage selects a set $\boldsymbol{W_2} \subset \boldsymbol{W_1}$ of detection windows, the features extracted during the first stage, $F_1(\boldsymbol{W_1})$, and the responses $R_1(\boldsymbol{W_1})$, are available for all detection windows, $\boldsymbol{W_1}$, considered by that stage. This

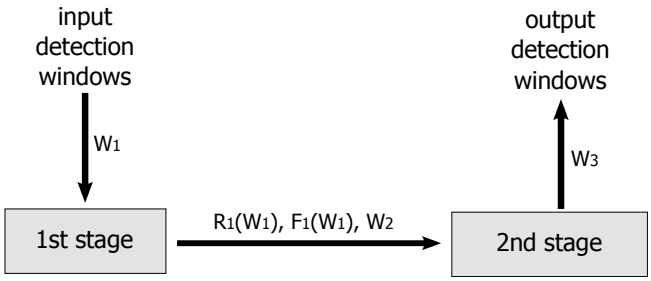


Figure 4.2: Steps of the PLS detector. Given an input set of detection windows, $\boldsymbol{W_1}$, the first stage outputs the windows selected to be considered by the second stage, $\boldsymbol{W_2} \subset \boldsymbol{W_1}$, the response and the features for all input detection windows, $R_1(\boldsymbol{W_1})$, $F_1(\boldsymbol{W_1})$, respectively.

information will be used by the regression models to avoid recomputing features for detection windows.

4.2.3 Data-Driven Detection Optimization Framework

As discussed earlier, our goal is to develop a framework to optimize detection methods by incorporating a set of online learned regression models based on cheap (and possibly already computed) feature descriptors to increase the number of detection windows rejected prior to expensive feature computation. This will lead to a significant reduction in computational cost while keeping accuracy high.

The regression models are used to accomplish two goals. First, they are used to estimate the generic detector's response at each stage, so that detection windows with low expected response can be quickly rejected. Second, based on the observation that multiple detection windows with high response but incorrect location tend to cluster around a true object location (as shown in Figure 4.5(b)), regression models are learned to estimate the correct location and size of objects. Both of

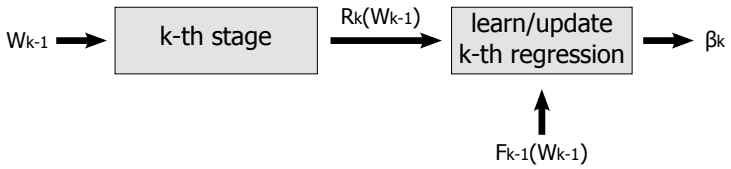


Figure 4.3: Learning a regression model to estimate the generic detector's responses for the k'st stage. Using features extracted from the previous stage, k-1, and responses obtained by the generic detector at the k'th stage, a regression model is learned using $F_{k-1}(\boldsymbol{W_{k-1}})$ and $R_k(\boldsymbol{W_{k-1}})$ as independent and dependent variables, respectively.

these regression models use features already available as the independent variables, i.e. features computed by the k-th detection stage are used to learn the regression used at the k+1'st stage. Therefore, overhead generated by feature extraction can be largely avoided.

4.2.3.1 Regression to Estimate the Detector's Response

Since the goal is to estimate the detector response without having to compute expensive features used by the generic detector for the k'th-stage, features from the previous stage are used as independent variables. The responses of the detector at the k'th-stage are considered as dependent variables. Figure 4.3 illustrates the learning process. The details of the model are described as follows.

We now describe how the detection model used to estimate the detector's response at each stage is learned. The approach is data-driven – the regression models are not learned from a training set but from the video sequence in which the detection is being performed. After applying the generic detection method to the first frame of the video sequence to obtain the detector's response and the features

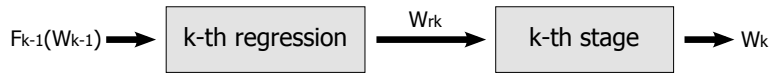


Figure 4.4: Execution of the regression model to estimate the generic detector's response for the k'th stage. Before executing the k'th stage, features extracted from the $k-1$'th stage are used to estimate the response of the generic detector at the k'th stage. The detection windows with low response are rejected before expensive features for the current stage need to be extracted.

extracted for each stage, an initial regression model is built to estimate each stage's responses.

Once the models to estimate each stage's responses are available, they are used to reject detection windows before the execution of the generic detector at a given stage, as illustrated in Figure 4.4. Note that $\boldsymbol{W_{rk}} \subset \boldsymbol{W_{k-1}}$. After the detector is executed, only the responses for detection windows in the set $\boldsymbol{W_{rk}}$ will be available, denoted by $R_k(\boldsymbol{W_{rk}})$. Therefore, the regression model for the k-th stage is rebuilt by adding $F_{k-1}(\boldsymbol{W_{rk}})$ and $R_k(\boldsymbol{W_{rk}})$ as independent and dependent variables, respectively. Practically, a list of limited size is considered to store samples used to build the regression model, and when new samples are added and the size of the list exceeds its limit, samples from older frames are discarded first. This allows the regression model to adapt to changes (i.e., changes in background patterns and statistics) that take place over time.

4.2.3.2 Regression to Estimate Object Location

In addition to the estimation of detector's response for each stage, we also use a regression model to predict the object location. This allows us to reduce further the number of detection windows that need to be considered by the generic detector at a given stage. For this regression model, while the independent variables are also the features extracted for the previous stage, the dependent variables are the tuple $(\Delta x, \Delta y, \Delta w, \Delta h)$, where Δx and Δy denote the difference on the x and y axes of the centroid of a given detection window and the correct location of the object, and Δw and Δh denote the difference between a given detection window's width and height and the correct object size, respectively.

Since ground truth object location is not known during the detection for a video sequence, the correct location used to learn the regression model is assumed to be the location specified by the last stage of the generic detector after performing non-maximum suppression.

The algorithm used to learn the regression model is as follows. After non-maximum suppression is performed on the detection windows selected by the last stage, only the ones with response higher than a threshold are stored in a set $\boldsymbol{W_t} = \{d_1, d_2, \ldots, d_k\}$ where $d_i = (cx_i, cy_i, w_i, h_i)$. Here, (cx_i, cy_i) denotes its centroid and w_i, h_i its width and height, respectively. Then, values for $(\Delta x, \Delta y, \Delta w, \Delta h)$ of each incorrect detection window considered by the last stage are computed and added with their respective features to learn the regression model. Once learned, the regression model is used to estimate the correct location of objects given a

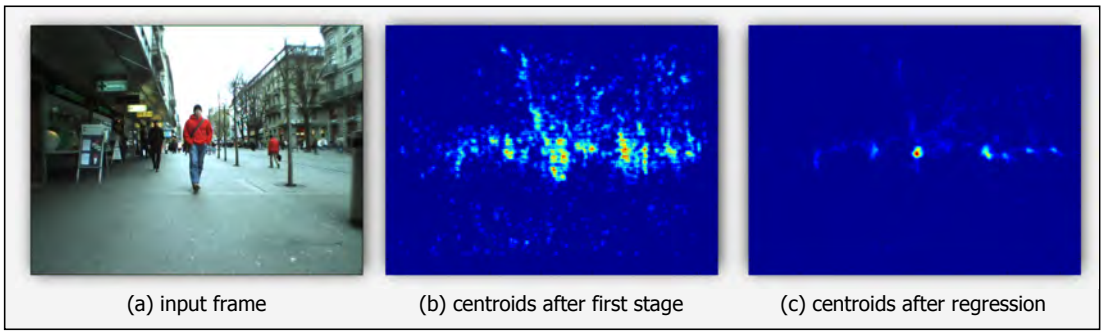


Figure 4.5: Location of objects. (a) input frame; (b) map showing the concentration of centroid locations for detection windows selected after the first stage of the PLS detector (regions in red show larger concentration than regions in blue); (c) concentration of the centroids after executing the regression to estimate object locations. The peaks are located mostly at the same positions as the pedestrians are in the input frame.

detection window's locations and its features. Similar to the regression models used to estimate detection responses, this regression model is also updated over time.

During execution, for a set of detection windows $\boldsymbol{W_s}$ selected from a previous stage, the features describing each detection window $d_i \in \boldsymbol{W_s}$ are projected using the regression model and the tuple $(\Delta x_i, \Delta y_i, \Delta w_i, \Delta h_i)$ is obtained. The new (and expected to be correct) location of d_i is then $(cx_i + \Delta x_i, cy_i + \Delta y_i, w_i + \Delta w_i, h_i + \Delta h_i)$. Then, considering the new locations of the detection windows, non-maximum suppression is conducted and the detection windows with small estimated response are rejected. Note that if non-maximum suppression were applied before correcting the locations, fewer detection windows would be rejected because they would be more sparsely located in the frame, as illustrated in Figure 4.5.

4.2.4 Integrating the Regression Models into the PLS Detector

Now we describe how the regressions models are integrated into the PLS detector. Figure 4.1 shows a flowchart of the resulting method, the black arrows describe the path of selected detection windows and the dashed-red arrows show the flow used to update the regression models. The steps are sorted in a increasing order of complexity (additional features are added along the path). Detection windows with low responses are rejected as early as possible to reduce the computational time significantly.

The integrated detector is described as follows. First we construct regression models to estimate the detector's response for each stage, the *response regression I* to estimate the responses of the first stage and the *response regression II* to estimate the responses of the second stage. Furthermore, a regression model to estimate the location of the objects, called *the location regression*, is added right after the response regression II so that additional detection windows can be rejected before reaching the computationally expensive second stage.

Features extracted during the first stage are used for the response regression II and the location regression, which avoids extracting features to be used specifically for these two regressions. However, features need to be computed to apply the response regression I since no features are available at this point of the process. Therefore, a feature extraction module is added prior to the first regression (denoted *simple features*). This module extracts a very small number of features for each input detection window (a subset of the features used in the first stage), which is used

by the response regression I to reject quickly a large number of detection windows. Although only a few features are used, experiments show that approximately 98% of the input detection windows can be rejected at the response regression I without decreasing detection accuracy when compared to the original PLS detector.

4.3 Experimental Results

In this section we describe the results obtained by the integrated detection method. After describing the experimental setup, we analyze how often the regression models need to be updated in order to provide a satisfactory trade-off between recall and computational cost. Then, we evaluate the detection results when regression models are learned offline and no updates are performed. Finally, we compare the detection results obtained by the original PLS detector with our proposed integrated detection method and analyze the rejection of detection windows and the speed-up obtained when each regression model is incorporated into the detection process.

4.3.1 Experimental Setup

To evaluate our method we use the ETHZ pedestrian dataset [17], which is composed of three video sequences collected from a moving platform. The first frame of each sequence is shown in Figure 4.6. These sequences contain frames of size 640×480 pixels. For all the experiments the detection is performed over 16 scales to consider humans with heights between 60 and 500 pixels, with strides of 4

Figure 4.6: First frame of each video sequence used to evaluate the proposed method.

pixels on the x-axis and 8 pixels on the y-axis. This setup results in 64, 292 detection windows per frame.

The features used for the integrated detector are the following. The features extracted for the first regression, referred to as F_1, are HOG features computed from blocks of 32×32 pixels with stride of 8 pixels. The features, F_2, extracted for the first stage are also HOG features computed from blocks of 16×16 and 32×32 pixels with strides of 8 and 16 pixels respectively. Finally, the features used for the second stage, F_3, are the same as those used by [62]. Note that $F_1 \subset F_2 \subset F_3$, therefore features computed earlier can be reused in later stages.

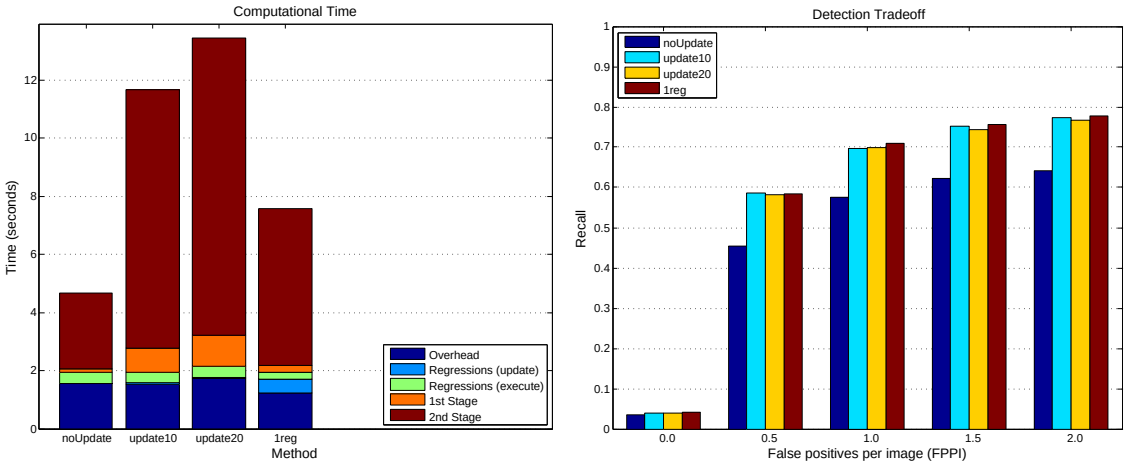
In the generic PLS detector, a threshold is used at each stage to reject detection windows presenting low responses (first stage uses 0.2 and second stage 1.0). The same thresholds used for the first and second stages are considered for the response regressions I and II, respectively. For the location regression a threshold based on the intersection-over-union measure [19] between detection windows is considered. If the intersection-over-union between two detection windows, after

correcting their location by the response, is greater than a threshold, the detection windows with lower (expected) response are rejected. In our experiments we use 0.5 as this threshold.

To evaluate several aspects of the integrated method and compare them to the original PLS detector, we consider the following detector setups. *Orig*: the generic PLS detector is applied; *1reg*: only the response regression I is incorporated with the generic method; *NMS*: after executing the response regression I and the first stage, non-maximum suppression is conducted before executing the second stage; *2reg*: the response regressions I and II are incorporated with the generic detector; *3reg*: the fully integrated method, as shown in Figure 4.1, is considered.

We conduct evaluations with respect to computational time, detection tradeoff, and the number of detection windows selected to be considered at each step of the detection process. In plots of computational time, as in Figure 4.11, the total time is divided into overhead (time to load the images and compute the integral histograms for HOG), updating and executing the regressions, and the first and second stages. The detection tradeoff is either shown for a single fixed value of false positive per image (FPPI) to evaluate the computational time, as in Figure 4.9, or for multiple values of FPPI, as in Figure 4.7(b). Finally, details regarding the number of detection windows selected after each step are shown in plots as the one in Figure 4.10.

All experiments were conducted on an Intel Core i7-860 processor, 2.8 GHz with 4GB of RAM running Windows 7 using a single processor core. The method was implemented in C++.



(a) (b)

Figure 4.7: Frequency of updates for the regression models. In *noUpdate* the response regression I is learned based on the first frame of the sequence and never updated, in *update10* and *update20* the models are updated every 10 and 20 frames, respectively, and in *1reg* the update is performed every frame. (a) average computational time per frame for each setup; (b) detection rates at fixed false positive per image.

4.3.2 Updating the Regression Models

Our first experiment evaluates the frequency with which the regression models are updated, a parameter for the method. For this experiment, we use video sequence #3 of the ETHZ dataset to setup this parameter and then we use it in the remaining experiments, where we consider all three videos. Figure 4.7(a) compares the average computational time per frame to the frequency that the response regression I is updated. We see that the fastest approach is obtained when no updates are performed (about 4.25s per frame), followed by the approaches that perform updates every frame (1reg), every 10 and 20 frames, respectively. In addition, according to the detection tradeoff shown in Figure 4.7(b), we see that the poorest detection result is obtained when the regression is not updated and the best when

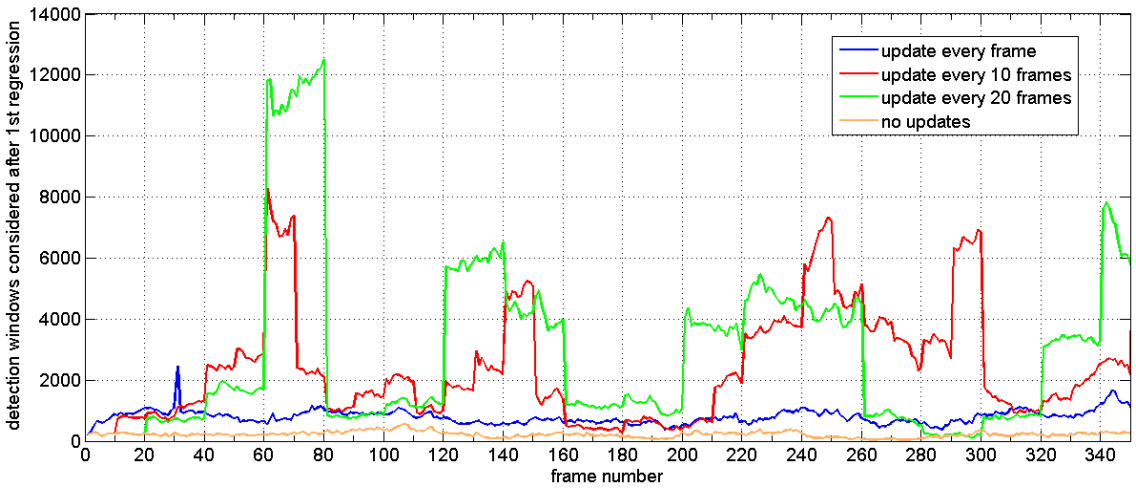


Figure 4.8: Number of detection windows selected by the response regression I for each frame for sequence #3.

the regression models are updated every frame.

Although the time consumed to update the regressions every frame is greater, as shown in Figure 4.7(a), the number of detection windows selected by the response regression I is smaller than the number of detection windows selected when the update is performed at every 10 or 20 frames, as shown in Figure 4.8. This makes the overall computational time smaller when the regressions are updated more often.

Based on the results shown in Figure 4.7, we conclude that the best trade-off between speed and detection results is achieved when the regressions are updated every frame. Therefore, in the remaining experiments the regression models will be updated every frame.

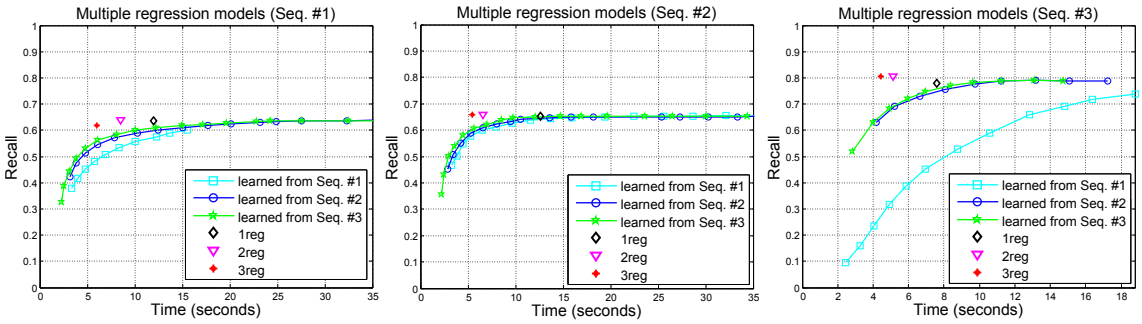


Figure 4.9: Comparison between online and offline learning of the regression models. All values for recall are obtained with FPPI fixed at 1.5. Online learning: 1reg, 2reg, and 3reg. Offline learning: learned from Seq. #1, learned from Seq. #2, and learned from Seq. #3.

4.3.3 Offline Learning of the Regression Models

This experiment aims at assessing the degradation of the detection rates when the regression models are learned offline using either a different video sequence or the same video sequence (in this latter case the regression models are not updated after few frames to emulate an offline learning).

Figure 4.9 compares online and offline learning for the regression models. The results obtained for the three video sequences clearly show that online learning provides higher recall for a fixed computational cost. In addition, we can see that even when a higher computational cost is allowed, the detection obtained with the offline learning does not outperform the results obtained by the online learning.

In none of the sequences does online learning provide lower recall than offline learning. However, depending on the video sequence that the offline regression models are learned from, the detection rates can be significantly degraded; i.e. when the regression models are learned from sequence #1 and the detection is performed

Table 4.1: Detection trade-off at different values of false positive per image.

Sequence	**FPPI**	**Recall**				
		Orig	1reg	NMS	2reg	3reg
Seq. #1	0.0	**0.086**	0.086	0.066	0.079	0.058
	0.5	0.399	0.397	0.344	**0.401**	0.357
	1.0	0.513	0.513	0.446	**0.518**	0.487
	1.5	0.592	0.591	0.497	**0.596**	0.574
	2.0	0.640	0.635	0.528	**0.638**	0.619
Seq. #2	0.0	0.008	0.008	0.008	**0.009**	0.007
	0.5	0.537	0.539	0.442	**0.544**	0.533
	1.0	0.612	0.613	0.488	**0.614**	0.609
	1.5	0.637	0.636	0.509	**0.642**	0.639
	2.0	0.652	0.652	0.521	**0.658**	0.657
Seq. #3	0.0	0.042	0.042	0.038	**0.042**	0.036
	0.5	0.572	0.583	0.491	**0.616**	0.586
	1.0	0.706	0.709	0.579	**0.753**	0.735
	1.5	0.755	0.755	0.616	**0.789**	0.787
	2.0	0.781	0.777	0.636	**0.805**	0.805

in sequence #3.

So, even though there is overhead due learning and applying the regressions, online learning provides consistently higher accuracy and lower computational cost than offline learning.

4.3.4 Evaluation and Comparisons

Since we are proposing a framework to speed up an existing detection method, we compare accuracy and speed between the original PLS detector and the integrated method described in Section 4.2.4. We also evaluate the individual modules.

Figures 4.10-4.15 compare several variations of the method, where we vary the number of detection windows considered by each module and the computational time for the three video sequences. Table 4.1 compares the recall obtained by each

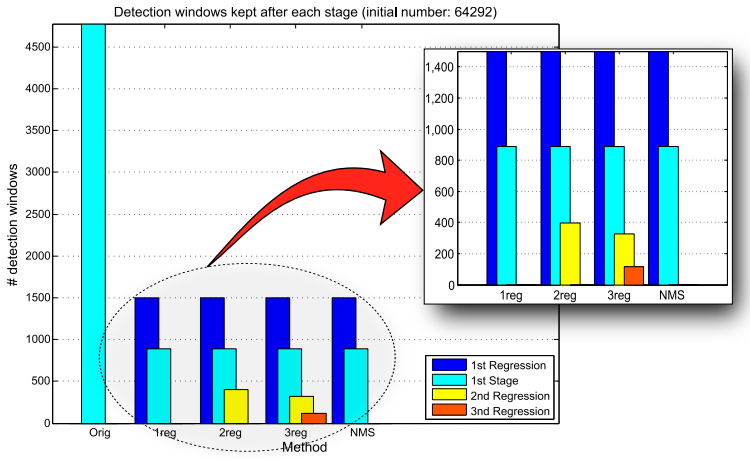


Figure 4.10: Number of detection windows considered for each part of the integrated method when detection is performed using sequence #1.

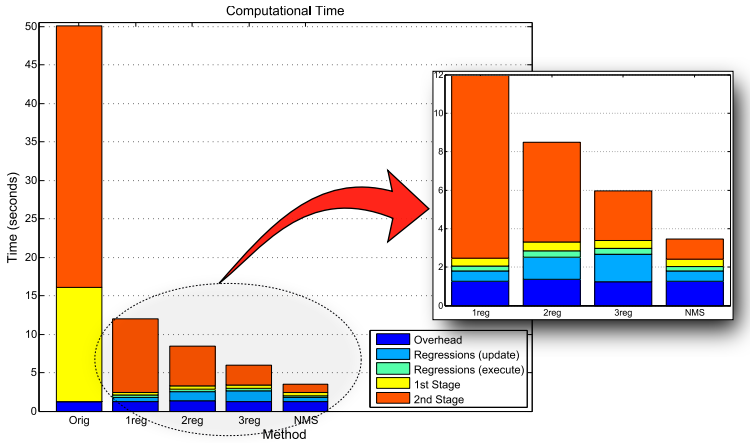


Figure 4.11: Computational time for each part of the integrated method when detection is performed using sequence #1.

setup when multiple false positive per image values are considered.

Regarding the number of detection windows, we see that on later steps of the detection process the number of detection windows selected is reduced significantly. It is important to observe that even though very few features are used in the response regression I, approximately 98% of the input detection windows are rejected at this

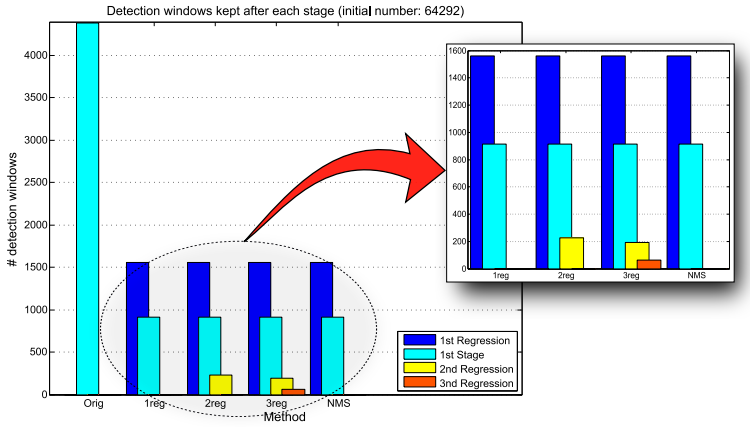


Figure 4.12: Number of detection windows considered for each part of the integrated method when detection is performed using sequence #2.

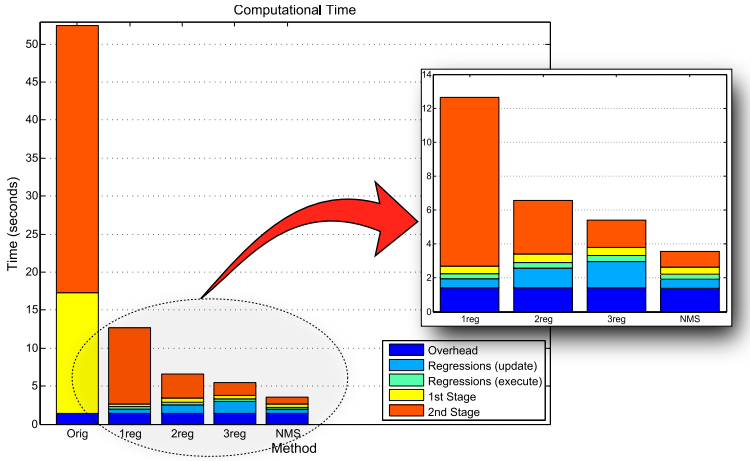


Figure 4.13: Computational time for each part of the integrated method when detection is performed using sequence #2.

step. This high rejection rate is due to the online learning, which provides a way of tuning the regressions to the environment being considered.

Based on the plots showing computational time, it is clear (and expected due to the large number of features) that the second stage has the highest computational cost - more than half of the time is spent in the second stage. This emphasizes the

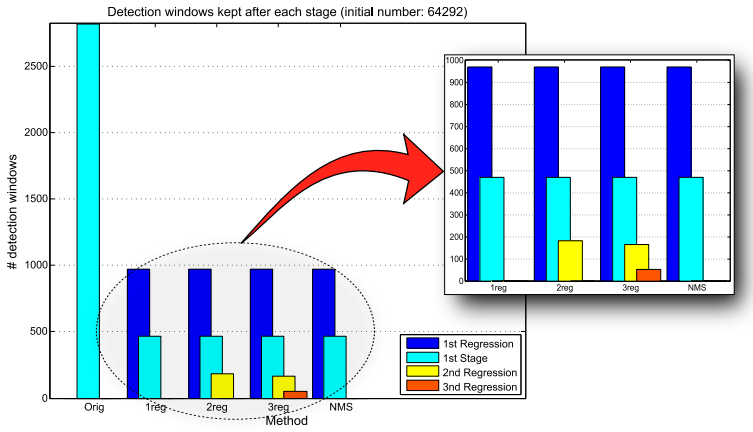


Figure 4.14: Number of detection windows considered for each part of the integrated method when detection is performed using sequence #3.

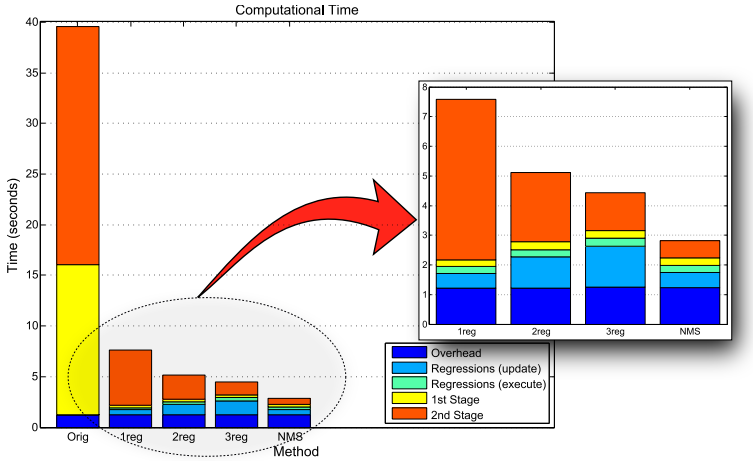


Figure 4.15: Computational time for each part of the integrated method when detection is performed using sequence #3.

importance of incorporating the regression models before this stage to reject as many detection windows as possible. Furthermore, we also see that there is some overhead added when more regression models are incorporated; this is mainly due to the time required to update the models, because the time to apply the regressions is almost constant and very small.

Table 4.2: Summary of the detection results. Recall is shown for FPPI fixed at 1.5.

Sequence		Orig	1reg	NMS	2reg	3reg
Seq. #1	time	50.1s	12.0s	3.45s	8.48s	5.97s
	recall	0.592	0.591	0.497	0.596	0.574
	speed-up	1.0×	4.1×	14.5×	5.9×	8.4×
Seq. #2	time	52.4s	12.6s	3.57s	6.57s	5.42s
	recall	0.637	0.636	0.508	0.642	0.639
	speed-up	1.0×	4.1×	14.6×	8.0×	9.6×
Seq. #3	time	39.5s	7.59s	2.81s	5.12s	4.43s
	recall	0.755	0.755	0.616	0.789	0.787
	speed-up	1.0×	5.2×	14.0×	7.7×	8.9×

Besides the speed-up achieved by incorporating the regressions, Table 4.1 shows that when the first two regression models are considered (2reg), the recall increases for all three sequences compared to the generic method. Therefore, even though many fewer detection windows are being considered by the second stage, the rejection of detection windows based on data-driven regressions is keeping only the windows more likely to be in their correct locations.

Table 4.2 summarizes the detection results showing the average computational time per frame, the speed-up obtained compared to the original PLS method, and the recall obtained when the FPPI is fixed at 1.5. We see that the highest speed-up is obtained by the method applying non-maximum suppression after the first stage, but the reduction in recall is unacceptable. The best trade-off between speed-up and recall is achieved by *2reg*. In addition, even though there is a slight drop in recall, the setup *3reg* achieves a significant speed-up, providing an acceptable trade-off between speed-up and recall when higher detection speeds are necessary.

Finally, the choice between *2reg* and *3reg* can be guided by the user's goal. If the accuracy can be slightly lower but speed is necessary, then *3reg* can be used; on the other hand, if accuracy is the most important factor, *2reg* should be considered. In any case, the original PLS detector, which is several times slower, does not need to be used since *2reg* provides higher recall.

Chapter 5

Learning Discriminative Appearance-Based Models

Appearance-based person recognition has widespread applications such as tracking and person identification and verification. However, the nature of the input data poses great challenges due to variations in illumination, shadows, and pose, as well as frequent inter- and intra-person occlusion. Under these conditions, the use of a single feature channel, such as color-based features, may not be powerful enough to capture subtle differences between different people's appearances. Therefore, additional cues need to be exploited and combined to improve discriminability of appearance-based models.

In general, human appearances are modeled using color-based features such as color histograms [12]. Spatial information can be added by representing appearances in joint color spatial spaces [16]. Also, appearance models of individuals based on nonparametric kernel density estimation have been used [37]. Other representations include spatial-temporal appearance modeling [23] and part-based appearance modeling [34].

Due to improvements that can be achieved using feature combination, we consider feature augmentation to model people's appearances. We augment color-based features with other discriminative cues. We exploit features based on textures and edges, obtaining a richer feature descriptor set as result.

To detect subtle differences between appearances, it is useful to perform a dense sampling for each feature channel, as will be shown on the experiments. However, as a result, the dimensionality of the feature space increases considerably (a feature vector describing an appearance is composed of more than 25,000 features).

Once discriminative appearance-based models have been built, machine learning methods need to be applied so that new samples of the appearances can be correctly classified during a testing stage. Learning methods such as support vector machines (SVM) [9], k-neareast neighbors combined with SVM [83], decision trees [4], learning discriminative distance metrics [37] have been exploited. However, since feature augmentation results in a high dimensional feature space, these machine learning methods may not always be used directly due to high computational requirements and low performance, as we show in the experimental results. The dimensionality of the data needs to be reduced first. We apply PLS for this.

The projection vectors estimated by PLS provide information regarding the importance of features as a function of location. Since PLS is a class-aware dimensionality reduction technique, the importance of features in a given location is related to the discriminability between appearances. For example, Figure 5.1 shows the spatial distribution of the weights of the first projection vector when PLS is used to combine the three feature channels. High weights are located in regions that better distinguish a specific appearance from the remaining ones. For example, blacks regions of the homogeneous jackets are not given high weights, since several people wear black jackets. However, the regions where the white and red jackets are located obtain high weights due to their unique appearances.

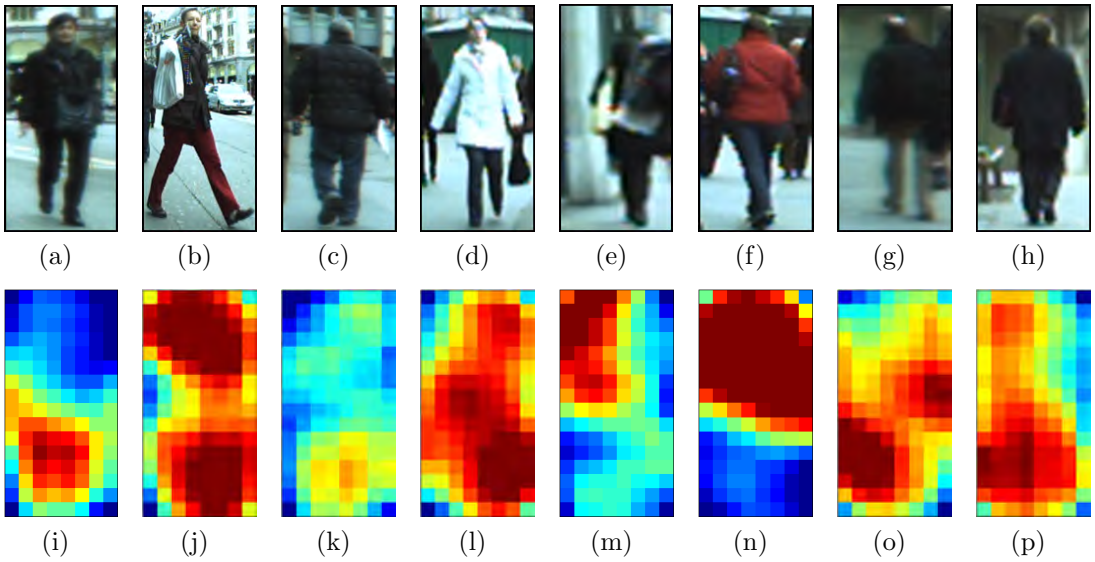


Figure 5.1: Spatial distribution of weights of the discriminative appearance-based models considering eight people extracted from video sequence #0 of the ETHZ dataset. The first row shows the appearance of each person and the second row the weights estimated by PLS for the corresponding appearance. Models are learned using the proposed method combining color, texture and edge features. PLS is used to reduce the dimensionality and the weights of the first projection vector are shown as the average of the feature weights in each block. Red indicates high weights, blue low.

Here we exploit a rich feature set analyzed by PLS using an one-against-all scheme [48] to learn discriminative appearance-based models. The dimensionality of the feature space is reduced by PLS and then a simple classification method is applied for each model using the resulting latent variables. This classifier is used during the testing stage to classify new samples. Experimental results based on appearance-based person recognition demonstrate that the feature augmentation provides better results than models based on a single feature channel. Additionally, experiments show that the proposed approach outperforms results obtained by techniques such as SVM and PCA.

5.1 Proposed Method

In this section we describe the method used to learn the appearance models. The features used are described in section 5.1.1 and section 5.1.2 describes the learning stage of the discriminative appearance-based models.

5.1.1 Feature Extraction

In the learning stage, only one exemplar is provided for each appearance i in the form of an image window. This window is decomposed into overlapping blocks and a set of features is extracted for each block to construct a feature vector. Therefore, for each appearance i, we obtain one sample described by a high dimensional feature vector $\boldsymbol{v}_i$.

To capture texture we extract features from co-occurrence matrices and edge information captured using HOG. In addition to these cues, color is also considered. In order to incorporate color we use color histograms computed for blocks. To avoid artifacts obtained by monotonic transformation in color and linear illumination changes, before calculating the histogram the value of pixels within a block are transformed to the relative ranks of intensities for each color channel R, G and B, similarly to [37]. Finally, each histogram is normalized to have unit L_2 norm.

Once the feature extraction process is performed for all blocks inside an image window, features are concatenated creating a high dimensional feature vector $\boldsymbol{v}_i$.

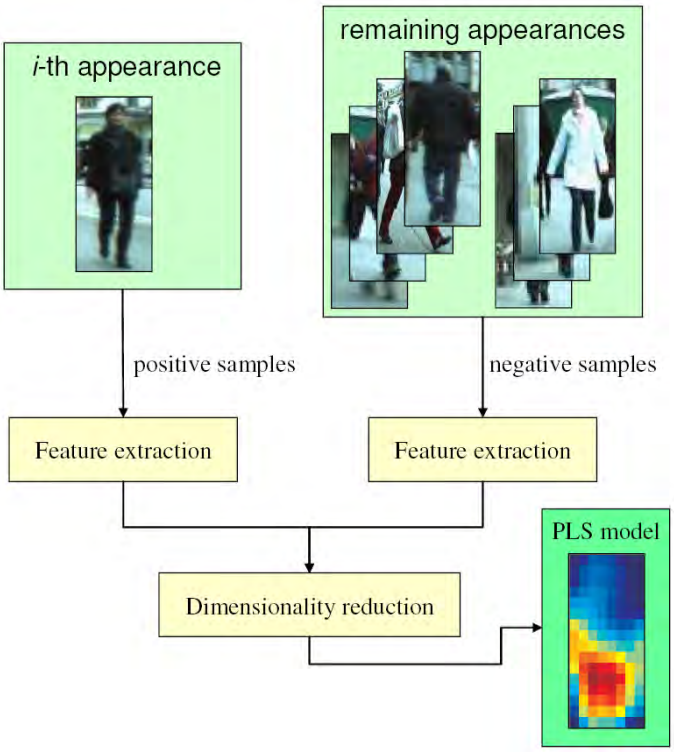


Figure 5.2: Proposed method. For each appearance represented by an image window, features are extracted and PLS is applied to reduce dimensionality using a one-against-all scheme. Afterwards, a simple classifier is used to match new samples to models learned.

5.1.2 Learning Appearance-Based Models

The procedure to learn the discriminative appearance-based models for a training set $t = \{\boldsymbol{u}_1, \boldsymbol{u}_2, \ldots, \boldsymbol{u}_k\}$, where $\boldsymbol{u}_i$ represents a subset of exemplars of each person (appearance) to be considered, is illustrated in Figure 5.2 and described in details as follows. Each subset $\boldsymbol{u}_i$ is composed of feature vectors extracted from image windows containing examples of the i-th appearance.

Here we exploit one-against-all scheme to learn a PLS discriminatory model for each person. Therefore, when the i-th person is considered, the remaining samples

$t \setminus \boldsymbol{u}_i$ are used as counter-examples of the i-th person.

For the one-against-all scheme, PLS gives higher weights to features located in regions containing discriminatory characteristics, as shown in Figure 5.1. Therefore, this process can be seen as a feature selection process depending on the feature type and the location.

Once the PLS model has been estimated for the i-th appearance, the feature vectors describing this appearance are projected onto the weight vectors. The resulting low-dimensional features are used during the testing stage to match a query samples.

When a sample is presented during the testing stage, its feature vector is projected onto the latent subspace estimated previously for each one of the k appearances and has its Euclidean distance to the samples used in training are computed. Then, this sample is classified as belonging to the appearance with the smallest Euclidean distance.

5.2 Experimental Results

In this section we present experiments to evaluate our approach. Initially, we describe the parameter settings and the dataset used. Then, we evaluate several aspects of our method, such as the improvement provided by using a richer feature set, the reduction in computational cost and improvement in performance compared to PCA and SVM.

Dataset. To obtain a large number of different people captured in uncon-

(a) sequence #1 (b) sequence #2 (c) sequence #3

Figure 5.3: Samples of the video sequences used in the experiments. (a) sequence #1 is composed of 1,000 frames with 83 different people; (b) sequence #2 is composed of 451 frames with 35 people; (c) sequence #3 is composed of 354 frames containing 28 people.

trolled conditions, we choose the ETHZ dataset [17] to perform our experiments. This dataset, originally used for human detection, is composed of four video sequences, where the first (sequence #0) is used to estimate parameters and the remaining three sequences are used for testing. Samples of testing sequence frames are shown in Figure 5.3.

The ETHZ dataset presents the desirable characteristic of being captured from moving cameras. This camera setup provides a range of variations in people's appearances. Figure 5.4 shows a few samples of a person's appearance extracted from different frames. Changes in pose and illumination conditions take place and due to the fact that the appearance model is learned from a single sample, a strong set of features becomes important to achieve robust appearance matching during the testing stage.

To evaluate our approach, we used the ground truth information regarding people's locations to extracted samples from each video (considering only people

Figure 5.4: Samples of a person's appearance in different frames of a video sequence belonging to ETHZ dataset.

with size higher than 60 pixels). Therefore, a set of samples is available for each different person in the video. The learning procedure presented in Section 5.1.2 is executed using one sample chosen randomly per person. Afterwards, the evaluation (appearance matching) considers the remaining samples.

Experimental Setup. To obtain the experimental results we have considered windows of 32×64 pixels. Therefore, either to learn or match an appearance, we rescale the person size to fit into a 32×64 window.

For co-occurrence feature extraction we use block sizes of 16×16 and 32×32 with shifts of 8 and 16 pixels, respectively, resulting in 70 blocks per detection window for each color band. We work in the HSV color space. For each color band, we create four co-occurrence matrices, one for each of the ($0°$, $45°$, $90°$, and $135°$) directions. The displacement considered is 1 pixel and each color band is quantized into 16 bins. The 12 descriptors mentioned earlier are then extracted from each co-occurrence matrix. This results in $10,080$ features.

We calculate HOG features considering blocks with sizes ranging from 12×12 to 32×64. In our configuration there are 326 blocks. As in [13], 36 features are

extracted from each block, resulting in a total of $11,736$ features.

The color histograms are computed from overlapping blocks of 32×32 and 16×16 pixels extracted from the image window. 16-bin histograms are computed for the R, G and B color bands, and then concatenated. The resulting number of features extracted by this method is $5,472$. Aggregating across all three feature channels, the feature vector describing each appearance contains $27,288$ elements.

To evaluate the approach described in Section 5.1.2, we compare the results to another well-know dimensionality reduction technique, PCA, and to SVM. With PCA, we first reduce the dimensionality of the feature vector and then we use the same classification approach described for PLS. However, with SVM the data is classified directly in the original feature space.

We consider four setups for the SVM: linear SVM with one-against-all scheme, linear multi-class SVM, kernel SVM with one-against-all scheme, and kernel multi-class SVM. A polynomial kernel with degree 3 is used. In the experiments we used the LIBSVM [10].

Since the high dimensionality of the feature space poses difficulties to compute the covariance matrix for PCA, we use a randomized PCA algorithm [59]. In addition, the classification for PCA uses the same scheme described in Section 5.1.2 for PLS, where a query sample is classified as belonging to the model presenting the smallest Euclidean distance in the low dimensional space.

Experimental results are reported in terms of the cumulative match characteristic (CMC) curves. These curves show the probability that a correct match is within the k-nearest candidates (in our experiments k varies from 1 to 7).

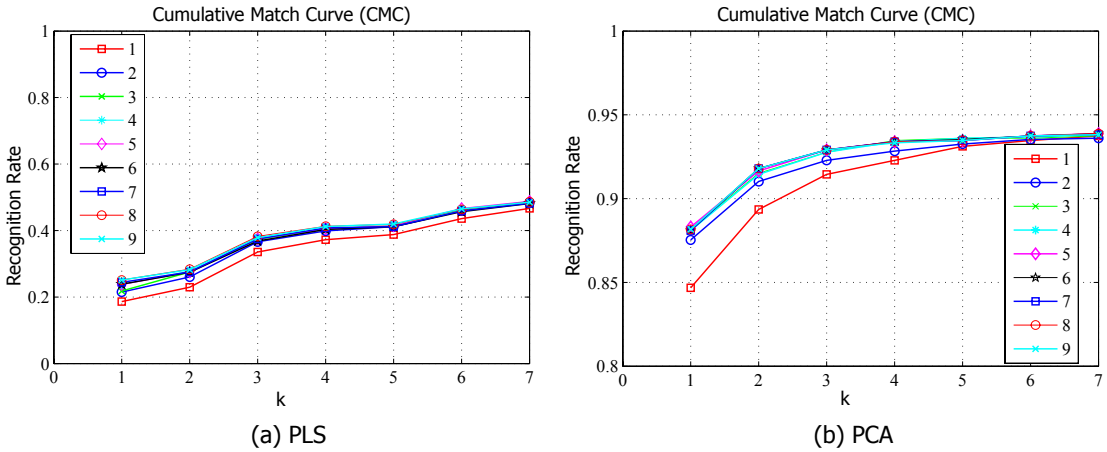


Figure 5.5: Recognition rate as a function of the number of factors (plots are shown in different scales to better visualization).

Before performing comparisons, we use the video sequence #0 to evaluate how many dimensions (number of weight vectors) should be used in the low dimensional latent space for PLS and PCA. Figure 5.5 shows the CMC curves for both when the number of factors is changed. The best results are obtained when 3 and 4 factors are considered for PLS and PCA, respectively. These parameters will be used throughout the experiments.

All experiments were conducted on an Intel Xeon, 3 GHz quad-core processor with 4GB of RAM running Linux operating system. The implementation is based on MATLAB.

Evaluation. Figures 5.6-5.8 show recognition rates obtained for each feature individually and their combination. In both cases the dimensionality is reduced using PLS. In general, the combination of features outperforms the results obtained when individual features are considered. This justifies the use of a rich set of features.

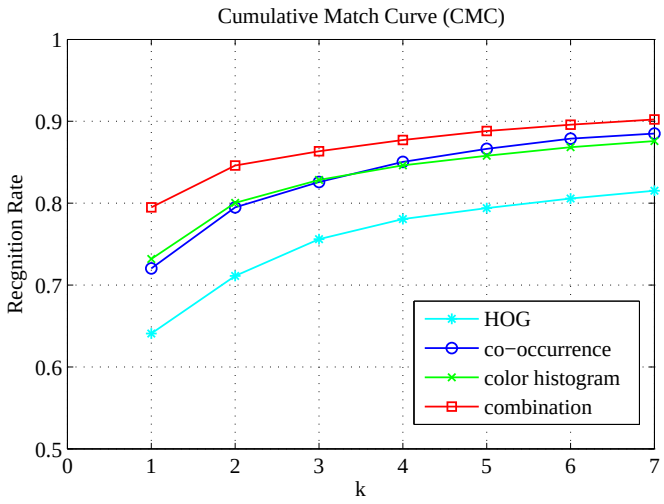


Figure 5.6: Recognition rates obtained by using individual features and combination of all three feature channels for sequence #1.

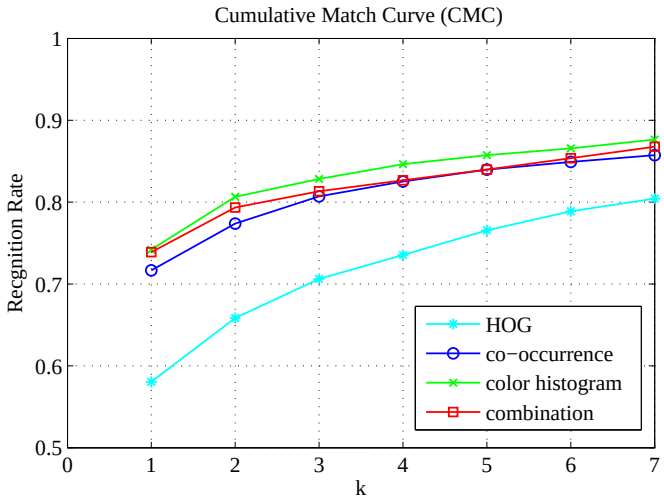


Figure 5.7: Recognition rates obtained by using individual features and combination of all three feature channels for sequence #2.

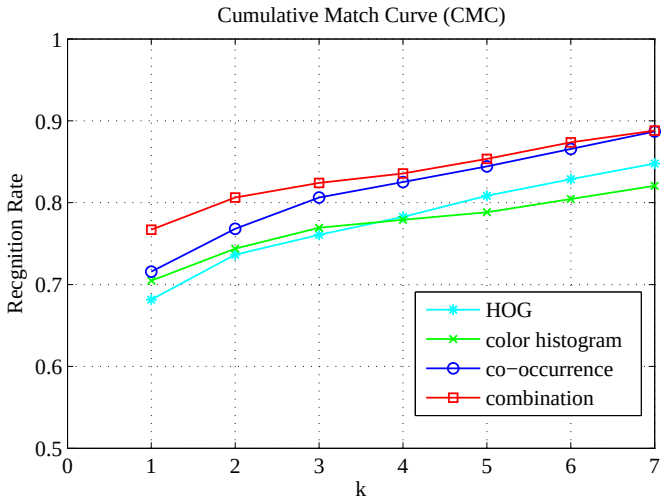
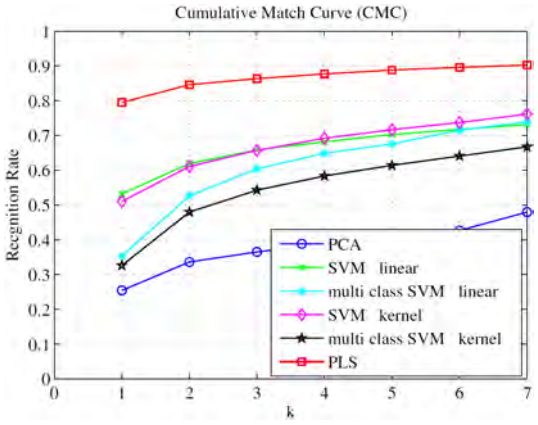


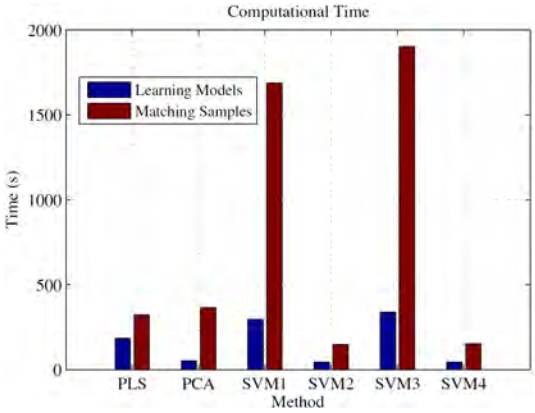
Figure 5.8: Recognition rates obtained by using individual features and combination of all three feature channels for sequence #3.

Figure 5.9 compares the PLS method to PCA and different setups of the SVM. We can see that the PLS approach obtains high recognition rates on the testing sequences of the ETHZ dataset. The results demonstrate, as one would expect, that PLS-based dimensionality reduction provides a more discriminative low dimensional latent space than PCA. In addition, we see that classification performed by SVM in high dimensional feature space when the number of training samples is small might lead to poor results. Finally, compared to the other methods, our approach achieves better results mainly when the number of different appearances being considered is high, i.e. sequences #1 and #2.

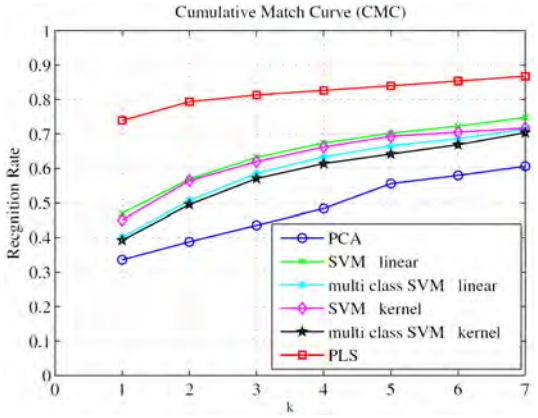
In terms of computational cost, Figure 5.9 shows that the proposed method, is in general, between PCA and SVM. The training and testing computational costs depend on the number of people and number of testing samples. Sequence #1 has $4,857$ testing samples amongst the 83 different people and sequences #2 and #3



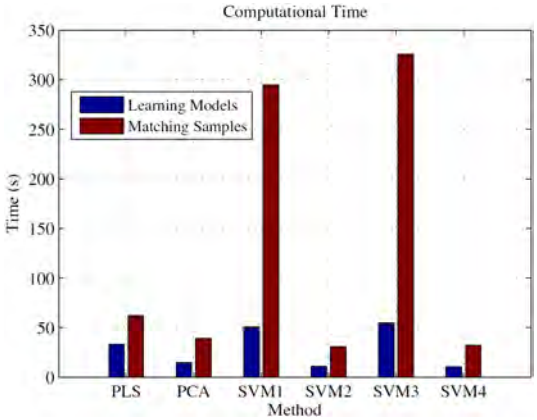
(a) Recognition rates for sequence #1



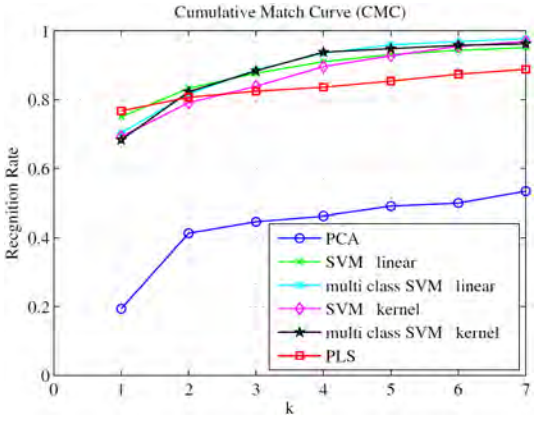
(b) Computational time for sequence #1



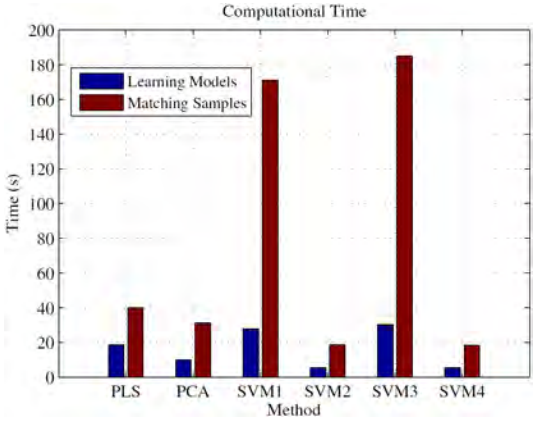
(c) Recognition rates for sequence #2



(d) Computational time for sequence #2



(e) Recognition rates for sequence #3



(f) Computational time for sequence #3

Figure 5.9: Performance and time comparisons considering the PLS method, PCA and SVM. SVM1: linear SVM (one-against-all), SVM2: linear SVM (multi-class), SVM3: kernel SVM (one-against-all), SVM4: kernel SVM (multi-class).

Figure 5.10: Misclassified samples of sequence #3. The images on the left show the training samples used to learn each appearance model. Images on the right contain samples misclassified by the PLS method.

have 1,961 and 1,762, respectively. The number of different people in each sequence is shown in Figure 5.3.

Figure 5.10 shows some of the misclassified samples of sequence #3 together with the samples used to learn the PLS models. We see that the misclassifications are due to changes in the appearance, occlusion and non-linear illumination change. This problem commonly happens when the appearance models are not updated over time. However, if integrated into a tracking framework, for example, the proposed method could use some model update scheme that might lead to higher recognition rates.

Finally, samples used to learn the appearance-based models for sequence #1 are shown in Figure 5.11. The large number of people and high similarity in their appearances increases the ambiguity among the models.

Figure 5.11: Samples of different people in sequence #1 used to learn the models.

Chapter 6

A Robust and Scalable Approach to Face Identification

The three primary face recognition tasks are *verification*, *identification*, and *watch list* [55]. In verification, the task is to accept or deny the identity claimed by a person. In identification, an image of an unknown person is matched to a gallery of known people. In the watch list task, a face recognition system must first detect if an individual is on the watch list. If the individual is on the watch list, the system must then correctly identify the individual. The method described here addresses the identification task.

Previous research has shown that face recognition under well controlled acquisition conditions is relatively mature and provides high recognition rates even when a large number of subjects is in the gallery [68, 87]. However, when images are collected under uncontrolled conditions, such as pose variations, uncontrolled lighting, and changes in facial expressions, the recognition rates decrease significantly.

Due to the large size of realistic galleries, not only the accuracy but also the scalability of a face identification system needs to be considered. The main scalability issues are the following. First, the number of subjects in the gallery can be quite large, so that common search techniques, such as brute force nearest neighbor, employed to match probe faces do not scale well. Second, in applications such as surveillance and human computer interaction, in which new subjects are

added incrementally, the necessity of rebuilding the gallery models every time a new subject is added compromises the computational performance of the system.

We tackle both problems. In order to reduce the problems associated with data collected under uncontrolled conditions, we consider a combination of low-level feature descriptors based on different clues. Then, feature weighting is performed by Partial Least Squares, which handles very high-dimensional data presenting multicollinearity and works well even when very few samples are available. Finally, a one-against-all classification scheme is used to model the subjects in the gallery.

To make the method scalable to the gallery size, we modify the one-against-all approach to use a tree-based structure. At each internal node of the tree, a binary classifier based on PLS regression is used to guide the search for the matching subject in the gallery. The use of this structure provides substantial reduction in the number of comparisons when a probe sample is matched against the gallery and also eliminates the need for rebuilding all PLS models when new subjects are added to the gallery.

6.1 Related Work

Detailed discussion of face recognition and processing can be found in recent and comprehensive surveys written by Tolba et al. [68] and Zhao et al. [87]. Most approaches to face recognition can be divided into two categories: holistic matching methods and local matching methods [89]. The methods in the former category use the whole face region to perform recognition and includes techniques such as sub-

space discriminant analysis, SVM, and AdaBoost; these may not cope well with the generalizability problem due to the unpredictable distribution of real-world testing face images. Probe images might be dramatically different from those considering during training [85]. The methods in the latter category first locate several facial features and then classify the faces according to local statistics.

Local binary patterns (LBP) and Gabor filters are descriptors widely used in face recognition. LBP is robust to illumination variations due to its invariance to monotonic gray-scale changes and Gabor filters are also robust to illumination variations since they detect amplitude-invariant spatial frequencies of pixel gray values [89]. There are several combinations or variations based on these descriptors that have been used for face recognition [3, 67, 85, 82].

Most recently developed face recognition systems work well when images are obtained under controlled conditions or when the test image is captured under similar conditions to those for the training images. However, under varying lighting or aging effects, their performance is still not satisfactory. To perform recognition under fairly uncontrolled conditions Tan and Triggs [66] proposed a preprocessing chain for illumination normalization. They used the local ternary patterns and a Hausdorff-like distance measure. Holappa [27] used local binary pattern texture features and proposed a filter optimization procedure for illumination normalization. Aggarwal [2] presented a physical model using Lambert's Law to generalize across varying situations. Shih [64] proposed a new color space LC_1C_2 as a linear transformation of the RGB color space.

Another challenge is that most current face recognition algorithms perform

well when several training images are available per subject; however they are still not adequate for scenarios where a single sample per subject is available. In real world applications, one training sample per subject presents advantages such as easy to collect galleries, low cost for storage and lower computational cost [65]. Thus, a robust face recognition system able to work with both single and several samples per subject is desirable. In [38], Liu et al. proposed representing each single (training, testing) image as a subspace spanned by synthesized shifted images and designed a new subspace distance metric.

Regarding the scalability issues discussed previously, there is also previous work focused on how to scale recognition systems to large datasets. In [81] a technique for combining rejection classifiers into a cascade is proposed to speed up the nearest neighbor search for face identification. Guo and Zhang [24] proposed the use of a constrained majority voting scheme for AdaBoost to reduce the number of comparisons needed.

6.2 Proposed Method

In this section, we first present the feature extraction process. Then, the proposed face identification approach is explained in two steps. Initially, we describe the one-against-all approach, then we describe the tree-based structure, which improves scalability when the gallery is large and reduces the computational cost of matching probe samples.

6.2.1 Feature Extraction

After cropping and resizing the faces, each sample is decomposed into overlapping blocks and a set of low-level feature descriptors is extracted from each block. The features used include information related to shape (captured by HOG), texture (captured by local binary pattern (LBP) [3]), and color information (captured simply by averaging the intensities of pixels in a block).

Local binary patterns [3] have been successfully applied in texture classification. LBP's characterize the spatial structure of the local image texture and are invariant under monotonic transformations of the pixel gray values. The LBP operator labels the pixels of an image by thresholding the 3×3 neighborhood of each pixel using the center value. A label is obtained by multiplication of the thresholded values by the binomial factors 2^p followed by their addition. The 256-bin histogram of the resulting labels is used as a feature descriptor.

Once the feature extraction process is performed for all blocks inside a cropped face, features are concatenated creating a high-dimensional feature vector $\boldsymbol{v}$. This vector is used to describe the face.

6.2.2 One-Against-All Approach

The procedure to learn models for subjects in the gallery $g = \{\boldsymbol{s}_1, \boldsymbol{s}_2, \ldots, \boldsymbol{s}_n\}$, where $\boldsymbol{s}_i$ represents exemplars of each subject's face, is illustrated in Figure 6.1 and described as follows. Each $\boldsymbol{s}_i$ is composed of feature vectors extracted from cropped faces containing examples of the i-th subject.

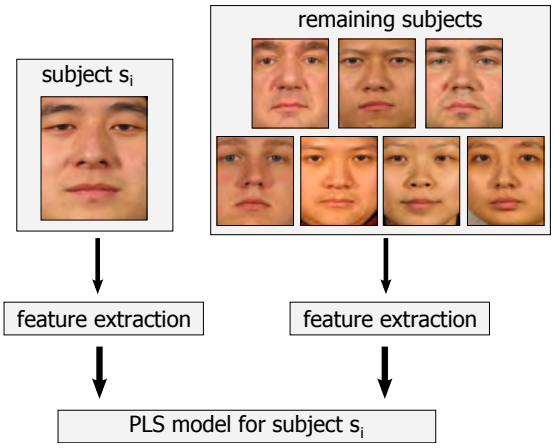


Figure 6.1: One-against-all face identification approach. Construction of the PLS regression model for a subject in the gallery.

As in Chapter 5, here we employ a one-against-all scheme to learn a PLS discriminatory model for each subject in the gallery. Therefore, when the i-th subject is considered, the remaining samples $g \setminus \boldsymbol{s}_i$ are used as counter-examples of the i-th subject. In addition, if the face dataset provides a training set we also add those samples, (excluding samples from the subject under consideration), as counter-examples of the i-th subject. Experiments show that the addition of training samples as counter-examples improves recognition rates.

Once the models have been estimated for all subjects in the gallery, the PLS regression models are stored to be later used to evaluate the responses for a probe sample. Then, when a probe sample is presented, its feature vector is projected onto each one of the PLS models. The model presenting the highest regression response gives the best match for the probe sample, as illustrated in Figure 6.2.

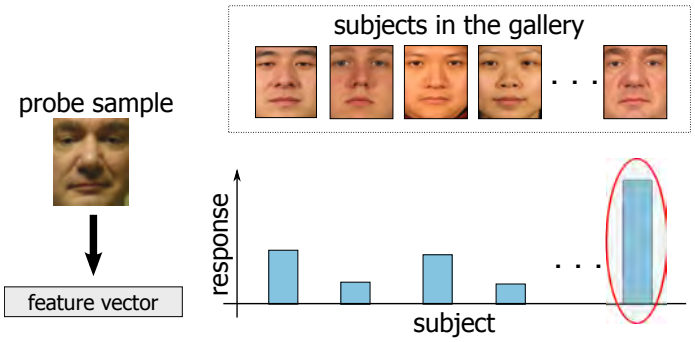


Figure 6.2: One-against-all face identification approach. Matching of a probe sample against the subjects in the gallery. The best match for a given probe sample is the one associated with the PLS model presenting the highest regression response.

6.2.3 Optimization Using a Tree-Based Structure

In terms of scalability, two drawbacks are present in the one-against-all scheme described in the previous section. First, when a new subject is added to the gallery, PLS models need to be rebuilt for all subjects. Second, to find the best match to a probe sample, the feature vector representing this sample needs to be projected onto all PLS models learned for the subjects in the gallery (common problem faced by methods that estimate matching scores using brute force nearest neighbor search [81]).

To reduce the need for projecting features onto all PLS models to find the best match for a probe sample, we construct a binary tree in which each node, n_j, contains a subset of the gallery subjects $t_j \subset g$, where $g = \{\boldsymbol{s}_1, \boldsymbol{s}_2, \ldots, \boldsymbol{s}_n\}$ as defined previously. A splitting procedure is used to decide which elements of t_j will belong to the left and right children of n_j, assigning at least one sample to each child. Each internal node is associated with a PLS regression model, used afterwards to guide

the search when probe samples are analyzed. In order to build the regression model for a node, the subjects assigned to the left child are defined to have response -1 and the subjects assigned to the right child are defined to have response $+1$. The splitting procedure and the building of PLS models are applied recursively in the tree until a node contains only a single subject (leaf node).

The application of the described procedure for a gallery with n subjects results in a tree containing n leaf nodes and $n-1$ PLS regression models located on the internal nodes.

We consider two approaches to split subjects between the children nodes. First, a procedure that uses PCA to create a low dimensional subspace (learned using samples from a training set) and then the K-means algorithm clusters data into two groups, each one is assigned to one child. The second approach chooses random splits and divides the subjects equally into two groups. We evaluate these splitting procedures in Section 6.3.3.

When a feature vector describing a probe sample is analyzed to find its best matching subject in the gallery, a search starting from the root of the tree is performed. At each internal node, the feature vector is projected onto the PLS model and according to its response, the search continues either from the left or from the right child. The search stops when a leaf node is reached. Figure 6.3 illustrates this procedure.

According to experimental results shown in Section 6.3.3, the traversal of a few search paths is enough to obtain the best match for a probe sample. Starting nodes for alternative search paths are stored in a priority queue. An internal node

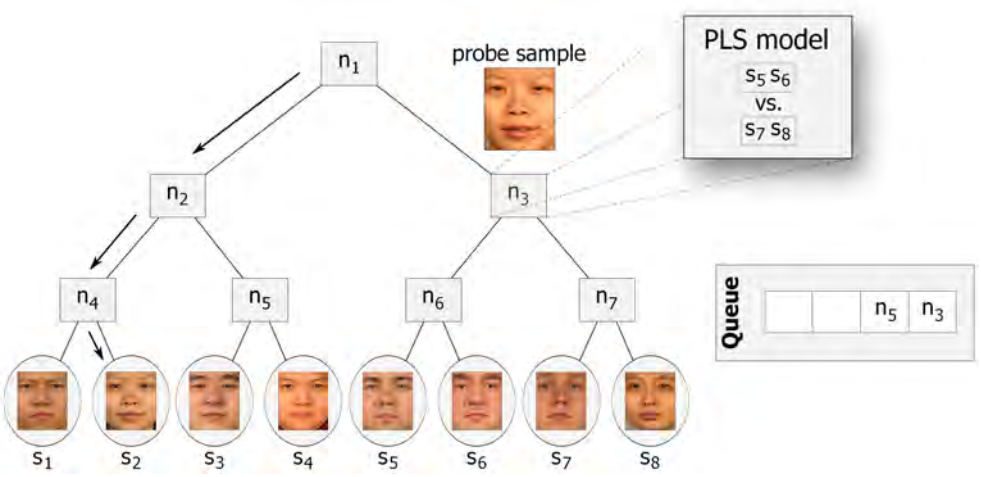


Figure 6.3: Tree-based structure used to optimize the search for matches to a probe sample. Each internal node contains a PLS regression model used to guide the search, as shown in details for node n_3, which has a PLS model constructed in a way that the response directs the search either to node n_6 or n_7. In this example the first path to be traversed is indicated by arrows (in this case, it leads to the correct match for this particular probe sample). Alternative search paths are obtained by adding nodes that have not been visited into a priority queue (in this example nodes n_3 and n_5 will be the starting nodes for additional search paths). After pursuing a number of search paths leading to different leaf nodes, the best match is chosen to be the one presenting the highest response (in absolute value).

n_k is pushed into the priority queue when its sibling is chosen to be in the current search path. The priority associated with n_k is proportional to its response returned by the PLS regression model at its parent. Finally, since each search path leads to a leaf node, the best match for a given probe sample is chosen to be the one presenting the highest response (in absolute values) among the leaf nodes reached during the search.

The tree-based structure can also be used to avoid rebuilding all PLS models when a new subject is added into the gallery. Assuming that a tree is built for k

subjects, the procedure to add a new subject $\boldsymbol{s}_{k+1}$ is described as follows. Choose a leaf node n_i, where $t_i = \{\boldsymbol{s}_j\}$; set n_i to be an internal node and create two new leaf nodes to store $\boldsymbol{s}_j$ and $\boldsymbol{s}_{k+1}$; then, build a PLS model for node n_i (now with $t_i = \{\boldsymbol{s}_j, \boldsymbol{s}_{k+1}\}$). Finally, rebuild all PLS models in nodes having n_i as a descendant. Therefore, using this procedure, the number of PLS models that needs to be rebuilt when a new subject is added no longer depends on the number of subjects in the gallery, but only on the depth of node n_i.

6.3 Experiments

In this section we evaluate several aspects of our proposed approach. Initially, we show that the use of the low-level feature descriptors analyzed by PLS in a one-against-all scheme, as described in Section 6.2.2, improves recognition rates over previous approaches, particularly when the data is acquired under uncontrolled conditions. Then, we demonstrate that the tree-based approach introduced in Section 6.2.3 obtains comparably high recognition rates with a significant reduction in the number of projections[1].

The method is evaluated on two standard datasets used for face recognition: FERET and FRGC version 1. The main characteristics of the FERET dataset are that it contains a large number of subjects in the gallery and the probe sets exploit differences in illumination, facial expression variations, and aging effects [57]. FRGC contains faces acquired under uncontrolled conditions [56].

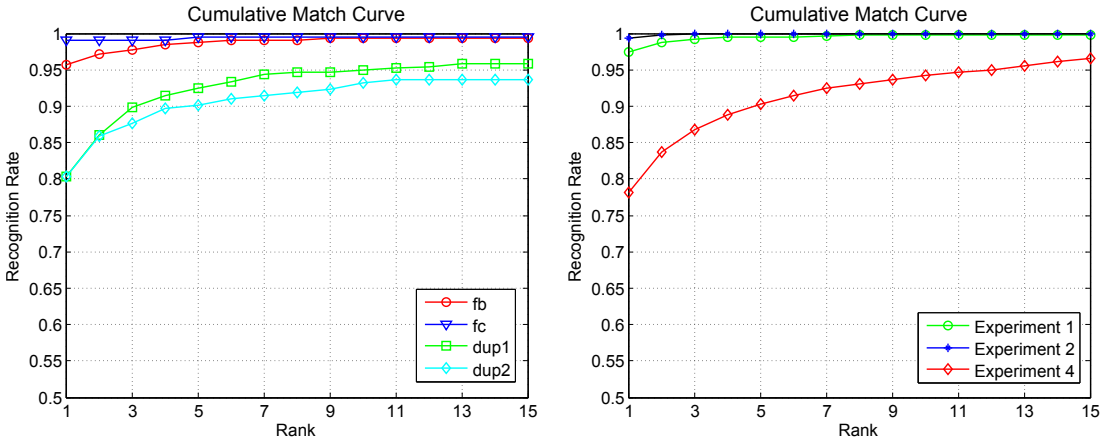
[1]The reduction on the number of projections is obtained due to reduction in the number of subjects in the gallery that need to be considered when probe samples are matched.

All experiments were conducted on an Intel Core i7-860 processor, 2.8 GHz with 4GB of RAM running Windows 7 operating system using a single processor core. The method was implemented using C++ programming language.

6.3.1 Evaluation on the FERET Dataset

The frontal faces in the FERET database are divided into five sets: fa (1196 images, used as gallery set containing one image per person), fb (1195 images, taken with different expressions), fc (194 images, taken under different lighting conditions), $dup1$ (722 images, taken at a later date), and $dup2$ (234 images, taken at least one year apart). Among these four standard probe sets, $dup1$ and $dup2$ are considered the most difficult since they are taken with time-gaps, so some facial features have changed. The images are cropped and rescaled to 110×110 pixels.

Experimental Setup. Since the FERET dataset is taken under varying illumination conditions, we preprocessed the images for illumination normalization. Among the best known illumination normalization methods are the self-quotient image (SQI) [73], total variation models, and anisotropic smoothing [27]. SQI is a retinex based method which does not require training images and has relatively low computational complexity; we use it due to its simplicity. Once the images are normalized, we perform feature extraction. For HOG features we use block sizes of 16×16 and 32×32 with strides of 4 and 8 pixels, respectively. For LBP features we use block size of 32×32 with a stride of 16 pixels. The mean features are computed from block size of 4×4 with stride of 2 pixels. This results in feature vectors with



(a) FERET (b) FRGC

Figure 6.4: The cumulative match curve for the top 15 matches obtained by the one-against-all approach based on PLS regression for FERET and FRGC datasets.

$35,680$ dimensions.

To evaluate how the method performs using information extracted exclusively from a single image per subject, in this experiment we do not add samples from the training set as counter-examples. The training set is commonly used to build a subspace to obtain a low dimensional representation of the features before performing the match. This subspace provides additional information regarding the domain of the problem.

Results and Comparisons. Figure 6.4(a) shows the cumulative match curves obtained by the one-against-all approach for all FERET probe sets. We see that our method is robust to facial expressions (fb), lighting (fc) and aging effect ($dup1$, $dup2$). The computational time to learn the gallery models is 4519 s and the average time to evaluate a pair of probe-gallery samples is 0.34 ms.

Table 6.1 shows the rank-1 recognition rates of previously published algorithms

Table 6.1: Recognition rates of the one-against-all proposed identification method compared to algorithms for the FERET probe sets.

	Method	**fb**	**fc**	**dup1**	**dup2**
using training set	Best result of [57]	95.0	82.0	59.0	52.0
	LBP [3]	97.0	79.0	66.0	64.0
	Tan [67]	98.0	98.0	90.0	85.0
not using training set	LGBPHS [85]	98.0	97.0	74.0	71.0
	HGPP [82]	97.6	98.9	77.7	76.1
	SIS [38]	91.0	90.0	68.0	68.0
	Ours	95.7	99.0	80.3	80.3

and ours on the FERET dataset. As shown in the table, the one-against-all approach achieves similar results on fb and fc without using the training set. Additionally, our results on the challenging $dup1$ and $dup2$ sets are over 80%.

6.3.2 Evaluation on the FRGC Dataset

We evaluate our method using three experiments of FRGC version 1 that consider 2D images. Experiment 1 contains a single controlled probe image and a gallery with one controlled still image per subject (183 training images, 152 gallery images, and 608 probe images). Experiment 2 considers identification of a person given a gallery with four controlled still images per subject (732 training images, 608 gallery images, and 2432 probe images). Finally, experiment 4 considers a single uncontrolled probe image and a gallery with one controlled still image per subject (366 training images, 152 gallery images, and 608 probe images). We strictly followed the published protocols. The images are cropped and rescaled to 275×320

pixels.

Experimental Setup. FRGC images are larger than FERET; thus we have chosen larger block sizes and strides to avoid computing too many features. For HOG features we use block sizes of 32×32 with strides of 8 pixels. For LBP features we use block size of 32×32 with strides of 24 pixels. And the mean features are extracted from block sizes of 8×8 with a stride of 4 pixels. This results in feature vectors with $86,634$ dimensions.

Experiment 4 in FRGC version 1 is considered the most challenging in this dataset. Since it is hard to recognize uncontrolled faces directly from the gallery set consisting of controlled images, we attempted to make additional use of the training set to create some *uncontrolled environment information* using morphed images. Morphing can generate images that with reduced resemblance to the imaged person or look-alikes of the imaged person [30]. The idea is to first compute a *mean face* from the uncontrolled images in the training set. Then, we perform triangulation-based morphing from the original gallery set to this mean face by 20%, 30%, 40%. This generates three synthesized images. Therefore, for each subject in the gallery we now have four samples.

Results and Comparisons. Figure 6.4(b) shows the cumulative match curves obtained by the one-against-all approach for the three probe sets of FRGC. In addition, the computational time to learn gallery models is 410.28 s for experiment 1, 1514.14 s for experiment 2, and 1114.39 s for experiment 4. The average time to evaluate a pair of probe-gallery samples is 0.61 ms.

Table 6.2 shows the rank-1 recognition rates of different algorithms on the

Table 6.2: Recognition rates of the one-against-all proposed identification method compared to other algorithms for the FRGC probe sets.

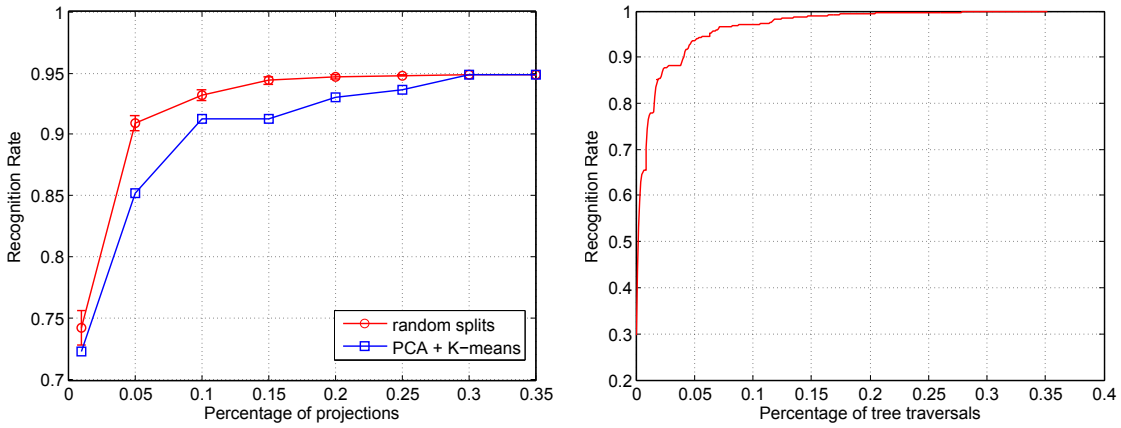
Method	**Exp.1**	**Exp.2**	**Exp.4**
PCA (from [41])	87.6	95.6	-
UMD [2]	94.2	99.3	-
LC_1C_2 [64]	-	-	75.0
Tan (from [27])	-	-	58.1
Holappa [27]	-	-	63.7
Ours	97.5	99.4	78.2

FRGC probe sets. Our method outperforms others in every probe set considered, especially on the most challenging experiment 4. This is, to the best of our knowledge, the best performance reported in the literature.

6.3.3 Evaluation of the Tree-Based Structure

In this section we evaluate the tree-based structure described in Section 6.2.3. First, we evaluate procedures used to split the set of subjects belonging to a node. Second, we test heuristics used to reduce the search space. Third, we compare the results obtained previously by the one-against-all approach to results obtained when the tree-based structure is incorporated. Finally, we compare our method to the approach proposed by Yuan et al. [81].

To evaluate the reduction in the number of comparisons, in this section the x-axis of the plots no longer displays the rank; instead it shows either the percentage of projections performed by the tree-based approach when compared to the one-



(a) (b)

Figure 6.5: Evaluation of the tree-based approach. (a) comparison of the recognition rates when random splits and PCA+K-Means approach are used; (b) evaluation of the heuristic based on stopping the search after a maximum number of tree traversals is reached.

against-all approach (e.g. Figure 6.5(a)) or the percentage of tree traversals when compared to the number of subjects in the gallery (e.g. Figure 6.5(b)). The y-axis displays the recognition rates for the rank-1 matches. We used probe set *fb* from the FERET dataset to perform evaluations in this section.

Procedure to Split Nodes. Figure 6.5(a) shows that both splitting procedures described in Section 6.2.3 obtain similar recognition rates when the same number of projections is performed. The error bars (in Figure 6.5(a)) show the standard deviation of the recognition rates obtained using random splits. They are very low and negligible when the percentage of projections increases. Due to the similarity of the results, we have chosen to split the nodes randomly. The advantages of applying random splits are the lower computational cost to build the gallery models and balanced trees are easily obtained. Balanced trees are important since

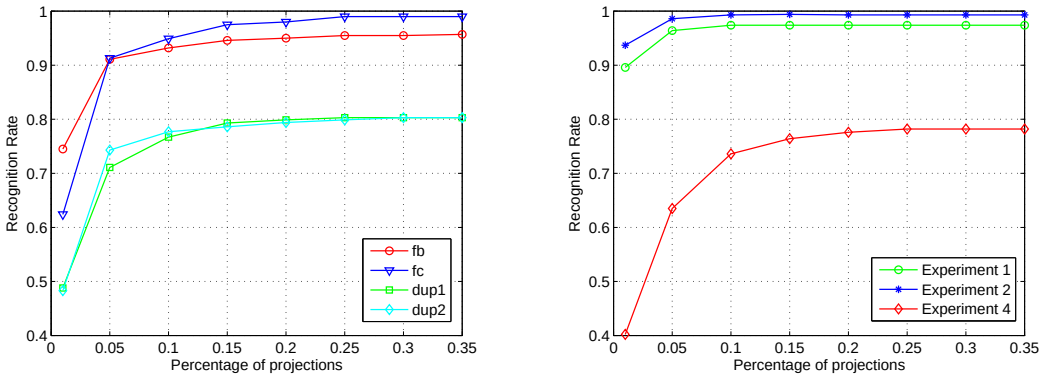
the depth of a leaf node is proportional to $\lg n$, which is desirable to keep short search paths.

Heuristics to Reduce the Search Space. The first experiment evaluates the recognition rate as a function of the maximum number of traversals allowed to find the match subject to a probe sample; this is limited to a percentage of the gallery size. Figure 6.5(b) shows the maximum recognition rates achievable for a given percentage. We can see that as low as 15% of traversals are enough to obtain recognition rates comparable to the results obtained by the one-against-all approach (95.7% for the probe set considered in this experiment).

In the second experiment we consider the following heuristic. For the initial few probe samples, all search paths are evaluated and the absolute values of the regression responses for the best matches are stored. The median of these values is computed. Then, for the remaining probe samples, the search is stopped when the regression response for a leaf node is higher than the estimated median value. Our experiments show that this heuristic alone is able to reduce the number of projections to 63% without any degradation in the recognition rates[2].

Results and Comparisons. Using the results obtained from the previous experiments (random splits and adding both heuristics to reduce the search space), we now compare the recognition rates obtained when the tree-based structure is used to results obtained by the one-against-all approach. Then, we evaluate the speed-up achieved by reducing the number of projections.

[2]Since the median is used, this heuristic is expected to work when more than 50% of the matches are correct. Therefore, it would fail only if the recognition rate for a dataset is lower than 50%.



(a) FERET (b) FRGC

Figure 6.6: Recognition rates as a function of the percentage of projections performed by the tree-based approach when compared to the one-against-all approach.

Figures 6.6(a) and 6.6(b) show identification results obtained for FERET and FRGC datasets, respectively. Overall, we see that when the number of projections required by the one-against-all approach is reduced to 20% or 30%, there is a negligible drop in the recognition rate shown in the previous sections. Therefore, without decreasing the recognition rate, the use of the tree-based structure provides a clear speed-up for performing the evaluation of the probe set. According to the plots, speed-ups of 4 times are achieved for FERET, and for FRGC the speed-up is up to 10 times depending on the experiment being considered.

Finally, we compare our method to the *cascade of rejection classifiers* (CRC) approach proposed by Yuan et al. [81]. Table 6.3 shows the speed-ups over the brute force nearest neighbor search and rank-1 error rates obtained by both approaches. We apply the same protocol used in [81] for the FRGC dataset. Higher speed-ups are obtained by our method and, differently from CRC, no significant increase in

Table 6.3: Comparison between our tree-based approach and the CRC approach.

	test set size as fraction of dataset	**10%**	**21%**	**32%**	**43%**	**65%**
CRC	speed-up	1.58	1.58	1.60	2.38	3.35
	rank-1 error rate	19.5%	22.3%	24.3%	28.7%	42.0%
Ours	speed-up	3.68	3.64	3.73	3.72	3.80
	rank-1 error rate	5.62%	5.08%	5.70%	5.54%	5.54%

the error rates is noticed when larger test set sizes are considered.

Conclusions

Characteristics presented by partial least squares such as selecting the most discriminative features for a given application, performing class-aware dimensionality reduction, and handling a large number of features in a fast manner provide a desirable framework for a wide range of applications in computer vision, especially for detection and recognition tasks.

We first proposed a human detection method, referred to as the PLS detector, using a rich descriptor set including edge-based features, texture measures and color information, obtaining a significant improvement in results over previously published approaches. The augmentation of these features generates a very high dimensional space where many classical machine learning methods are intractable. The characteristics of our data make an ideal setting for applying PLS to obtain a much lower dimensional subspace where we use simple and efficient classifiers. We have tested our approach on a number of varied datasets, demonstrated its good generalization capabilities and shown it to outperform state-of-the-art detection methods. Then we have described an extension of the PLS detector to handle partial occlusions. It combines face and person detectors into different models, and makes decisions based on the hypotheses derived from those models. Finally, we have proposed a set of data-driven regression models to estimate detector's responses and object locations using efficiently computable features. When integrated with the PLS detector, significant speed-up was obtained. The online learning performed on specific scenes not only provided speed-up, but also improvements on detection accuracy.

Second, we described a framework to learn discriminative appearance-based

models based on PLS. The results show that this method outperforms other approaches considering a one-against-all scheme. It has also been demonstrated that the use of a richer set of features leads to improvements in results. Then, also using the one-against-all scheme we have proposed a face identification method using a set of low-level feature descriptors analyzed by PLS which presents the advantages of being both robust and scalable. Experimental results have shown that the method works well for single image per sample, in large galleries, and under different conditions, such as variation in illumination, aging effect, and expression variations. The use of PLS regression makes the evaluation of probe-gallery samples very fast due to the necessity of only a single dot product evaluation. Optimization is further improved by incorporating the tree-based structure, which significantly reduces the number of projections when compared to the one-against-all approach, with negligible effect on recognition rates.

Bibliography

[1] Yotam Abramson, Bruno Steux, and Hicham Ghorayeb. Yet even faster (yef); real-time object detection. *Int. J. Intell. Syst. Technol. Appl.*, 2(2/3):102–112, 2007.

[2] G. Aggarwal, S. Biswas, and R. Chellappa. UMD experiments with FRGC data. In *CVPR Workshop*, pages 172–178, 2005.

[3] T. Ahonen, A. Hadid, and M. Pietikainen. Face recognition with local binary patterns. In *ECCV*, 35, pages 469–481, 2004.

[4] Y. Amit, D. Geman, and K. Wilder. Joint Induction of Shape Features and Tree Classifiers. *PAMI*, 19(11):1300–1305, 1997.

[5] J. Begard, N. Allezard, and P. Sayd. Real-time human detection in urban scenes: Local descriptors and classifiers selection with adaboost-like algorithms. In *CVPR Workshops*, 2008.

[6] P.N. Belhumeur, J.P. Hespanha, and D.J. Kriegman. Eigenfaces vs. Fisherfaces: Recognition Using Class Specific Linear Projection. *PAMI*, pages 711–720, 1997.

[7] S. Belongie, J. Malik, and J. Puzicha. Matching shapes. In *Computer Vision, 2001. ICCV 2001. Proceedings. Eighth IEEE International Conference on*, volume 1, pages 454–461 vol.1, 2001.

[8] S. Belongie, J. Malik, and J. Puzicha. Shape matching and object recognition using shape contexts. *IEEE Trans. Pattern Anal. Mach. Intell.*, 24(4):509–522, 2002.

[9] A. Bosch, A. Zisserman, and X. Muoz. Image Classification using Random Forests and Ferns. In *ICCV*, pages 1–8, 2007.

[10] Chih-Chung Chang and Chih-Jen Lin. *LIBSVM: a library for support vector machines*, 2001. Software available at www.csie.ntu.edu.tw/ cjlin/libsvm.

[11] Yu-Ting Chen and Chu-Song Chen. Fast human detection using a novel boosted cascading structure with meta stages. *Image Processing, IEEE Trans. on*, 17(8):1452–1464, 2008.

[12] D. Comaniciu, V. Ramesh, and P. Meer. Kernel-based Object Tracking. *PAMI*, 25(5):564–577, 2003.

[13] N. Dalal and B. Triggs. Histograms of Oriented Gradients for Human Detection. In *CVPR 2005*, pages 886–893, 2005.

[14] P. Dollar, B. Babenko, S.J. Belongie, P. Perona, and Z.W. Tu. Multiple Component Learning for Object Detection. In *ECCV 2008*, pages 211–224, 2008.

[15] Lars Elden. Partial least-squares vs. Lanczos bidiagonalization–I: analysis of a projection method for multiple regression. *Computational Statistics & Data Analysis*, 46(1):11 – 31, 2004.

[16] A. Elgammal, R. Duraiswami, and L.S. Davis. Probabilistic Tracking in Joint Feature-Spatial Spaces. In *CVPR*, volume 1, pages 781–788, 2003.

[17] A. Ess, B. Leibe, and L. Van Gool. Depth and appearance for mobile scene analysis. In *ICCV*, October 2007.

[18] A. Ess, B. Leibe, K. Schindler, and L. Gool. A mobile vision system for robust multi-person tracking. *CVPR*, 2008.

[19] Mark Everingham et al. The 2005 pascal visual object classes challenge. In *Selected Proceedings of the 1st PASCAL Challange Workshop*, 2006.

[20] P. Felzenszwalb, D. McAllester, and D. Ramanan. A discriminatively trained, multiscale, deformable part model. In *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*, pages 1–8, June 2008.

[21] Yoav Freund and Robert E. Schapire. A decision-theoretic generalization of online learning and an application to boosting. In *EuroCOLT '95: Proceedings of the Second European Conference on Computational Learning Theory*, pages 23–37, London, UK, 1995. Springer-Verlag.

[22] D.M. Gavrila. The visual analysis of human movement: A survey. *Computer Vision and Image Understanding*, 73(1):82–98, 1999.

[23] Niloofar Gheissari, Thomas B. Sebastian, and Richard Hartley. Person Reidentification Using Spatiotemporal Appearance. In *CVPR*, pages 1528–1535, 2006.

[24] Guo-Dong Guo and Hong-Jiang Zhang. Boosting for fast face recognition. In *ICCV Workshop*, Los Alamitos, CA, USA, 2001. IEEE Computer Society.

[25] Robert M. Haralick, K. Shanmugam, and Dinstein. Textural features for image classification. *Systems, Man and Cybernetics, IEEE Transactions on*, 3(6):610–621, 1973.

[26] B. Heisele, T. Serre, and T. Poggio. A Component-based Framework for Face Detection and Identification. *IJCV*, 74(2):167–181, 2007.

[27] J. Holappa, T. Ahonen, and M. Pietikinen. An optimized illumination normalization method for face recognition. In *IEEE International Conference on Biometrics: Theory, Applications and Systems*, pages 6–11, 2008.

[28] R.L. Hsu, M. Abdel-Mottaleb, and AK Jain. Face detection in color images. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 24(5):696–706, 2002.

[29] O. Javed, S. Ali, and M. Shah. Online detection and classification of moving objects using progressively improving detectors. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, volume 1, pages 696 – 701 vol. 1, june 2005.

[30] B. Kamgar Parsi, E. Lawson, and P. Baker. Toward a human-like approach to face recognition. In *BTAS*, pages 1–6, 2007.

[31] Yan Ke and R. Sukthankar. Pca-sift: a more distinctive representation for local image descriptors. In *Computer Vision and Pattern Recognition, 2004. CVPR 2004. Proceedings of the 2004 IEEE Computer Society Conference on*, volume 2, pages 506–513, June-2 July 2004.

[32] A. Kembhavi, B. Siddiquie, R. Miezianko, S. McCloskey, and L.S. Davis. Scene it or not? incremental multiple kernel learning for object detection. *International Conference on Computer Vision*, 2009.

[33] Svetlana Lazebnik, Cordelia Schmid, and Jean Ponce. Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories. In *CVPR '06: Proceedings of the 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 2169–2178, Washington, DC, USA, 2006. IEEE Computer Society.

[34] Jian Li, S.K. Zhou, and R. Chellappa. Appearance Modeling Under Geometric Context. In *ICCV*, volume 2, pages 1252–1259, 2005.

[35] Li-Jia Li, Gang Wang, and Li Fei-Fei. Optimol: automatic online picture collection via incremental model learning. In *Computer Vision and Pattern Recognition, 2007. CVPR '07. IEEE Conference on*, pages 1 –8, june 2007.

[36] Z. Lin and L. S. Davis. A pose-invariant descriptor for human detection and segmentation. In *ECCV*, 2008.

[37] Zhe Lin and Larry S. Davis. Learning Pairwise Dissimilarity Profiles for Appearance Recognition in Visual Surveillance. In *International Symposium on Advances in Visual Computing*, pages 23–34, 2008.

[38] J. Liu, S. Chen, Z. Zhou, and X. Tan. Single image subspace for face recognition. In *IEEE International Workshop on Analysis and Modeling of Faces and Gestures*, pages 205–219, 2007.

[39] David G. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60:91–110, 2004.

[40] S. Maji, A.C. Berg, and J. Malik. Classification using intersection kernel support vector machines is efficient. In *CVPR*, June 2008.

[41] A. Mian, M. Bennamoun, and R. Owens. 2D and 3D multimodal hybrid face recognition. In *ECCV*, volume 3, pages 344–355, 2006.

[42] Krystian Mikolajczyk, Cordelia Schmid, and Andrew Zisserman. Human detection based on a probabilistic assembly of robust part detectors. In *ECCV 2004*, volume I, pages 69–81, 2004.

[43] A. Mohan, C. Papageorgiou, and T. Poggio. Example-based object detection in images by components. *PAMI*, 23(4):349–361, 2001.

[44] H. Moon, R. Chellappa, and A. Rosenfeld. Optimal edge-based shape detection. *Image Processing, IEEE Transactions on*, 11(11):1209–1227, 2002.

[45] Fabien Moutarde, Bogdan Stanciulescu, and Amaury Breheret. Real-time visual detection of vehicles and pedestrians with new efficient adaBoost features. In *2nd Workshop on Planning, Perception and Navigation for Intelligent Vehicles (PPNIV), at 2008 IEEE International Conference on Intelligent RObots Systems (IROS 2008)*, Nice France, 2008.

[46] Yadong Mu, Shuicheng Yan, Yi Liu, T. Huang, and Bingfeng Zhou. Discriminative local binary patterns for human detection in personal album. In *CVPR 2008*, pages 1–8, June 2008.

[47] S. Munder and D. Gavrila. An experimental study on pedestrian classification. *PAMI*, 28(11):1863–1868, 2006.

[48] Chikahito Nakajima, Massimiliano Pontil, Bernd Heisele, and Tomaso Poggio. Full-body Person Recognition System. *Pattern Recognition*, 36(9):1997–2006, 2003.

[49] Timo Ojala, Matti Pietikinen, and David Harwood. A comparative study of texture measures with classification based on featured distributions. *Pattern Recognition*, 29(1):51 – 59, 1996.

[50] E. Osuna, R. Freund, and F. Girosit. Training support vector machines: an application to face detection. In *CVPR*, pages 130–136, 1997.

[51] G. Overett, L. Petersson, L. Andersson, and N. Pettersson. Boosting a heterogeneous pool of fast hog features for pedestrian and sign detection. In *Intelligent Vehicles Symposium, 2009 IEEE*, pages 584 –590, june 2009.

[52] S. Paisitkriangkrai, Chunhua Shen, and Jian Zhang. Fast pedestrian detection using a cascade of boosted covariance features. *Circuits and Systems for Video Technology, IEEE Transactions on*, 18(8):1140 –1151, aug. 2008.

[53] Marco Pedersoli, Jordi Gonzàlez, and Juan José Villanueva. High-speed human detection using a multiresolution cascade of histograms of oriented gradients. In *IbPRIA '09: Proceedings of the 4th Iberian Conference on Pattern Recognition and Image Analysis*, pages 48–55, Berlin, Heidelberg, 2009. Springer-Verlag.

[54] N. Pettersson, L. Petersson, and L. Andersson. The histogram feature - a resource-efficient weak classifier. In *Intelligent Vehicles Symposium, 2008 IEEE*, pages 678 –683, june 2008.

[55] P. J. Phillips, P. J. Micheals, R. J. Blackburn, D. M. Tabassi, and J. M. Bone. Face Recognition vendor test 2002: Evaluation Report. Technical report, NIST, 2003.

[56] P. Jonathon Phillips, Patrick J. Flynn, Todd Scruggs, Kevin W. Bowyer, Jin Chang, Kevin Hoffman, Joe Marques, Jaesik Min, and William Worek. Overview of the face recognition grand challenge. In *CVPR*, pages 947–954, 2005.

[57] P. Jonathon Phillips, Hyeonjoon Moon, Syed A. Rizvi, and Patrick J. Rauss. The FERET evaluation methodology for face-recognition algorithms. *TPAMI*, 22:1090–1104, 2000.

[58] Victor Prisacariu and Ian Reid. fasthog - A real-time gpu implementation of hog. Technical Report 2310/09, Department of Engineering Science, Oxford University, 2009.

[59] V. Rokhlin, A. Szlam, and M. Tygert. A Randomized Algorithm for Principal Component Analysis. *ArXiv e-prints*, 2008.

[60] R. Rosipal and N. Kramer. Overview and recent advances in partial least squares. *Lecture Notes in Computer Science*, 3940:34–51, 2006.

[61] HA Rowley, S. Baluja, and T. Kanade. Neural network-based face detection. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 20(1):23–38, 1998.

[62] W. R. Schwartz, A. Kembhavi, D. Harwood, and L. S. Davis. Human detection using partial least squares analysis. In *ICCV*, 2009.

[63] V.D. Shet, J. Neuman, V. Ramesh, and L.S. Davis. Bilattice-based logical reasoning for human detection. In *CVPR*, 2007.

[64] P. Shih and C. Liu. Evolving effective color features for improving FRGC baseline performance. In *CVPR Workshop*, pages 156–163, 2005.

[65] X. Tan, S. Chen, Z. Zhou, and F. Zhang. Face recognition from a single image per person: A survey. *Pattern Recognition*, 39:1725–1745, 2006.

[66] X. Tan and B. Triggs. Enhanced local texture feature sets for face recognition under difficult lighting conditions. In *IEEE International Workshop on Analysis and Modeling of Faces and Gestures*, pages 168–182, 2007.

[67] Xiaoyang Tan and Bill Triggs. Fusing Gabor and LBP feature sets for kernel-based face recognition. In *AMFG*, pages 235–249, 2007.

[68] A.S. Tolba, A.H. El-Baz, and A.A. El-Harby. Face recognition: A literature review. *International Journal of Signal Processing*, 2:88–103, 2006.

[69] Duan Tran and David Forsyth. Configuration estimates improve pedestrian finding. In *NIPS 2007*, pages 1529–1536. MIT Press, Cambridge, MA, 2008.

[70] O. Tuzel, F. Porikli, and P. Meer. Human detection via classification on riemannian manifolds. In *CVPR*, 2007.

[71] P. Viola and M. Jones. Rapid object detection using a boosted cascade of simple features. In *Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on*, volume 1, pages I–511 – I–518 vol.1, 2001.

[72] P. Viola and M.J. Jones. Robust Real-Time Face Detection. *International Journal of Computer Vision*, 57(2):137–154, 2004.

[73] H. Wang, S. Z. Li, and Y. Wang. Face recognition under varying lighting conditions using self quotient image. In *IEEE International Conference on Automatic Face and Gesture Recognition*, pages 819–824, 2004.

[74] Christian Wojek, Gyuri Dorkó, André Schulz, and Bernt Schiele. Sliding-windows for rapid object class localization: A parallel technique. In *Proceedings of the 30th DAGM symposium on Pattern Recognition*, pages 71–81, Berlin, Heidelberg, 2008. Springer-Verlag.

[75] H. Wold. Partial least squares. In S. Kotz and N.L. Johnson, editors, *Encyclopedia of Statistical Sciences*, volume 6, pages 581–591. Wiley, New York, 1985.

[76] S. Wold, W. Johansson, and M. Cocchi. PLS - Partial Least-Squares Projections to Latent Structures. In H. Kubinyi, editor, *3D QSAR in Drug Design: Volume 1: Theory Methods and Applications*, pages 523–550. Springer Verlag, 1993.

[77] Bo Wu and R. Nevatia. Improving part based object detection by unsupervised, online boosting. In *Computer Vision and Pattern Recognition, 2007. CVPR '07. IEEE Conference on*, pages 1 –8, june 2007.

[78] Bo Wu and R. Nevatia. Optimizing discrimination-efficiency tradeoff in integrating heterogeneous local features for object detection. In *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*, pages 1–8, June 2008.

[79] Bo Wu and Ram Nevatia. Detection of multiple, partially occluded humans in a single image by bayesian combination of edgelet part detectors. In *Proceedings of the Tenth IEEE International Conference on Computer Vision*, pages 90–97, Washington, DC, USA, 2005. IEEE Computer Society.

[80] M.H. Yang, D.J. Kriegman, and N. Ahuja. Detecting Faces in Images: A Survey. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, pages 34–58, 2002.

[81] Quan Yuan, Ashwin Thangali, and Stan Sclaroff. Face identification by a cascade of rejection classifiers. In *CVPR Workshop*, pages 152–159, 2005.

[82] B. Zhang, S.G. Shan, X.L. Chen, and W. Gao. Histogram of gabor phase patterns (HGPP): A novel object representation approach for face recognition. *IEEE Transactions on Image Processing*, 16:57–68, 2007.

[83] Hao Zhang, A.C. Berg, M. Maire, and J. Malik. SVM-KNN: Discriminative Nearest Neighbor Classification for Visual Category Recognition. In *CVPR*, volume 2, pages 2126–2136, 2006.

[84] Li Zhang and R. Nevatia. Efficient scan-window based object detection using gpgpu. In *Computer Vision and Pattern Recognition Workshops, 2008. CVPRW '08. IEEE Computer Society Conference on*, pages 1 –7, june 2008.

[85] 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. In *ICCV'05*, pages 786–791, 2005.

[86] Wei Zhang, G. Zelinsky, and D. Samaras. Real-time accurate object detection using multiple resolutions. In *ICCV*, 2007.

[87] W. Zhao, R. Chellappa, P. J. Phillips, and A. Rosenfeld. Face recognition: A literature survey. *ACM Comput. Surv.*, 35(4):399–458, 2003.

[88] Qiang Zhu, Mei-Chen Yeh, Kwang-Ting Cheng, and S. Avidan. Fast human detection using a cascade of histograms of oriented gradients. In *CVPR 2006*, pages 1491–1498, 2006.

[89] J. Zou, Q. Ji, and G. Nagy. A comparative study of local matching approach for face recognition. *IEEE Transactions on Image Processing*, 16:2617–2628, 2007.