Face and Landmark Detection by Using Cascade of Classifiers

Hakan Cevikalp
Eskisehir Osmangazi University
Eskisehir, Turkey
hakan.cevikalp@gmail.com

Bill Triggs
Laboratoire Jean Kuntzmann
Grenoble Cedex 9, France
Bill.Triggs@imag.fr

Vojtech Franc
Czech Technical University
Praha, Czech Republic
xfrancv@cmp.felk.cvut.cz

***Abstract*— In this paper, we consider face detection along with facial landmark localization inspired by the recent studies showing that incorporating object parts improves the detection accuracy. To this end, we train roots and parts detectors where the roots detector returns candidate image regions that cover the entire face, and the parts detector searches for the landmark locations within the candidate region. We use a cascade of binary and one-class type classifiers for the roots detection and SVM like learning algorithm for the parts detection. Our proposed face detector outperforms the most of the successful face detection algorithms in the literature and gives the second best result on all tested challenging face detection databases. Experimental results show that including parts improves the detection performance when face images are large and the details of eyes and mouth are clearly visible, but does not introduce any improvement when the images are small.**

I. INTRODUCTION

In recent years, face detection has been thoroughly studied due to its wide potential applications, including face recognition, human-computer interaction, video surveillance, etc. Especially, in the context of face recognition, the detection of faces along with the detection of some fiducial points, such as the eyes and mouth, is the first step of the face recognition system, and this step largely affects the performance of the overall system.

In general, face detection can be defined as follows: Given an image, determine the presence of faces in the image and return the location and extent of each face. This is a challenging task since there are various factors that affect the appearance of faces such as variations in illumination, poses and facial expressions, occlusion, make-up, beard, mustache, glasses, etc. In addition to these factors, the intra-class variations among faces that arise from variations among size and shape of facial features make the face detection problem even harder.

Based on a survey [21] on face detection, existing face detection approaches are grouped into four categories: knowledge-based methods, template-based methods, feature invariant methods, and appearance-based methods. Among them, appearance-based methods distinguished themselves as the most promising ones, and we consider the appearance-based face detection in our study. There are mainly two important factors that determine the success of a face detector: the features used for representing face images and the learning algorithm that implements the detection. Early face detection methods used raw pixel values [15], wavelets [14], Gabor filter based features [16], etc., for image representation. Recently, histogram based features have become very popular owing to their successful performance and efficiency. Among these, local binary patterns (LBPs) [1], local ternary patterns (LTPs) [17], local edge orientation histograms [13], histograms of oriented gradients (HOGs) [5], [4] are worth mentioning. The most of the recent state of the art face detection methods usually use a combination of different features, rather than using one single representation, by just concatenating them or by optimizing combination coefficients at the learning stage [4].

Regarding the learning methodology, most approaches treat the face detection problem as a binary classification problem, namely, determining if the current detector window contains a correctly framed face instance, or anything else (background, a partial of incorrectly framed instance, etc.). Various machine learning methods ranging from the nearest neighbor classifiers to more complex approaches such as neural networks [15], convolution neural networks [9], and classification trees [6] have been used as classifier for face detection. However, two methods have received a great deal of attention owing to their interesting properties: boosting based cascades [13], [19] and the Support Vector Machines (SVMs) [20], [12]. Viola&Jones [19] introduced a very efficient face detector by using AdaBoost to train a cascade of pattern-rejection classifiers over rectangular wavelet features. Each stage of the cascade is designed to reject a considerable fraction of the negative cases that survive to that stage, so most of the windows that do not contain faces are rejected early in the cascade with comparatively little computation. As the cascade progresses, rejection typically gets harder so the single-stage classifiers grow in complexity. Although this method gives good results for real-time face detection, under less stringent time constraints SVM classifiers are currently becoming more popular [20], [12], [23]. Linear SVMs are usually preferred for their simplicity and speed, although it is well-established that kernel SVMs typically give higher accuracy at the cost of increased computational complexity. Therefore, recent general object and face detectors use short cascades in which the early stages use linear SVMs to reject most of the negative windows quickly, while the latter stages use nonlinear SVMs to make the final decisions. In addition to the binary classifiers, recent methods use one-class type classifiers for learning [4], [10]. The main idea is to focus on face class and to approximate the region spanned by the face image samples in the feature space. During detection, each window is assigned to face class or background based on the distances to the approximated face class model. To

this end, Jin et al. [10] used kernelized hypersphere model to approximate the face class region. Although this gives an accurate approximation to the class boundaries, it is computationally expensive since one needs to evaluate kernel functions against the returned support vectors. To speed up, the authors first divided the putative face window into 9 blocks using heuristic rules such as the eye regions are darker than the cheeks and the bridge of nose, etc., and applied the kernelized classifier if the region passes all tests. In a more recent study, we [4] use linear hyperplane and linear/kernelized hypersphere models in a cascade.

In this study, we focus on both face detection and facial fiducial landmark localization at the same time. These two tasks have traditionally been approached as separable problems - e.g., landmark localization is performed on images returned by a face detector. However, recent studies on general object detection revealed that incorporating parts during detection helps to capture object class spatial layout better and it significantly improves the detection accuracy [7], [22], [23]. Motivated by this, we learn face appearances as well as locations and deformations of facial landmarks from the training data. During detection, we first find the candidate image windows that include the entire face region by using the roots detectors and then search for the face parts in this region by using the parts detector. The final score is obtained by combining the confidence values coming from these two tasks. As a learning algorithm for the roots detector, we combine both binary and one-class type classifiers in a cascade structure. We use linear SVMs as binary classifier and use hyperplane and hypersphere models for one-class classification. Our linear hyperplane fitting algorithm differs from the one we introduced in [4] in the sense that it is more robust to outliers and yields better generalization performance. For the parts detectors, we use an SVM like learning algorithm that enforces negativity constraints on the weights of the separating hyperplane normal. In addition to these contributions, we also propose a novel constrained clustering algorithm based on convex modeling to distinguish between different face appearances.

II. METHOD

We train the roots and parts detectors to find the location of faces and face parts. Root detectors return candidate image windows that cover entire face region, and the parts detectors search for the face parts such as eyes and mouth within the candidate windows returned by the roots detectors. The responses of the roots and parts filters are computed at different resolutions and they are fused (by multiplying the scores of the roots and parts detectors) to yield a final score for each root location. We assign the local window as face class based on the output of the final score on the root location. In case of disagreements - such as detectors return conflicting overlapping root regions, we assign the root location corresponding to the highest score as face class and ignore the other conflicting detection. We now explain the details of training of the roots and parts detectors.

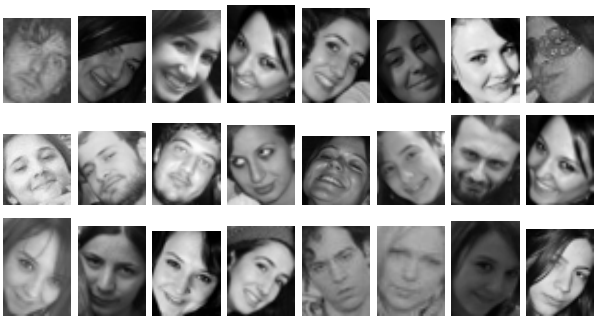

Fig. 1. Some of the tilted faces collected from real-world images.

A. Roots Detectors

Although frontal face images are considered as rigid objects, faces coming from real-world images may be tilted to the right or left as shown in Fig. 1. Therefore we train two roots that are symmetric along the vertical axis. To this end, we take the mirror of each positive face image and extract their features. As visual features, we use LBP+HOG features. Both LBP and HOG features are extracted by using a grid of 6×6 pixel cells. For LBP, features are created by using circular (8,1) neighborhoods, and the resulting histograms were normalized to sum 1. For HOG, we use 9 bins of unsigned gradient orientation over color images as in [7]. Then, resulting face visual features are clustered into two clusters: one cluster for right-tilted faces and the other for left-tilted faces by using the convex hull clustering algorithm defined below.

Clustering Based on Convex Hull Approximation: We have pairs of face feature vectors extracted from symmetric face images, and each element of any pair must be assigned to one of the two clusters. This is a constrained clustering problem and we use a bottom-up clustering approach for this. We first select the pair where the Euclidean distance between the feature vectors is the maximum and assign them to different clusters. Then we iteratively assign the examples of each pair to the nearest cluster. During finding the distances from any sample to the nearest cluster, we approximate each cluster by using the convex hull of the samples within that cluster, and find the minimum Euclidean distance from the current sample to each convex hull. In order to circumvent any problem arising from the overlapping convex hulls, we compute the distances from samples to the reduced convex hulls created by limiting the convex combination coefficients as in [2]. The nearest convex hull distance measure is more reliable than the nearest-mean or nearest-neighbor assignment in high-dimensional spaces as shown in [3]. This procedure is repeated until all pairs of examples are assigned to clusters. Then, we refine cluster assignments by randomly selecting pairs and re-computing the distances from the samples to the nearest clusters.

1) Cascade of Classifiers: Once we split the face features into two clusters, we train a root detector for each split by using a cascade of classifiers. Our cascade has four stages. The first stage is a linear SVM classifier and it efficiently

rejects as many of the background samples while keeping almost all face samples to the next stage. The second stage is a one-class type classifier that approximates the positive data by a hyperplane. This stage is also computationally very efficient and it passes almost all face examples while rejecting some of the background samples that passed the first stage. The third stage is a linear SVDD (Support Vector Data Description) classifier [18], which is based on a hypersphere model in the (non-kernelized) input space. This is also quite fast and it works in a complementary way with the previous cascades by rejecting most of the false positives that passed the first two stages. The last stage of the cascade makes the final decision, and it uses a nonlinear (kernelized) hyperpshere model to approximate the object class. This is slower, but it operates only on a small number of positives and difficult negatives passed the previous stages. We now present each of these stages in details by omitting the well-known linear SVM classifier.

Linear Hyperplane Approximation: The second stage of the cascade approximates the face class by an hyperplane and tests the distance of a given sample feature to this hyperplane. If the distance is between some pre-defined thresholds we consider it as a face sample and a background sample otherwise. To this end, we find an hyperplane that fits the face samples best and at the same time far from the negative samples at least by a predefined margin. Let $\mathbf{x}$ be the sample's feature vector and let $\mathbf{w}^\top \mathbf{x} + b = 0$ be the equation of the hyperplane. In that case, finding the best fitting discriminative hyperplane problem can be formulated as

$$\begin{aligned} \underset{\mathbf{w},\boldsymbol{\xi}\geq 0}{\arg\min}\ & \frac{1}{2}||\mathbf{w}||^2 + C_+ \sum_i (\xi_i + \xi_i^*) + C_- \sum_j \xi_j \\ \text{s.t. }\ & \mathbf{w}^\top \mathbf{x}_i + b \leq \Delta + \xi_i, \\ & \mathbf{w}^\top \mathbf{x}_i + b \geq -\Delta - \xi_i^*, \\ & \mathbf{w}^\top \mathbf{x}_j + b \geq \Delta + 1 - \xi_j,\ \ i \in I_+,\ \ j \in I_-. \end{aligned} \tag{1}$$

Here $C_+(C_-)$ is a user defined parameter that controls the weight of the errors associated to the positive (negative) samples, $I_+(I_-)$ is the set including indices of positive (negative) samples, and Δ is a constant that can be set a value between 0 and 1. By this formulation, we are constraining positive face samples to lie between two parallel hyperplanes $\mathbf{w}^\top \mathbf{x} + b = \Delta$ and $\mathbf{w}^\top \mathbf{x} + b = -\Delta$. The negative samples lie to the right of the hyperplane $\mathbf{w}^\top \mathbf{x} + b = 1 + \Delta$ separated from positive samples by at least margin $1/||\mathbf{w}||$ (A better approach would be to let negative samples lie on both sides of the hyperplane, but this problem is not convex and hard to solve for large scale data.) Positive slack variables, ξ_i, ξ_i^*, ξ_j, are introduced for the samples violating the constraints. This is a convex-quadratic programming problem that can be solved by any quadratic program solver. In high-dimensional spaces, this formulation gives a more robust fitting compared to the least-squares based fitting formulation described in [4].

Linear Hypersphere Approximation: The third stage of the cascade consists of a single linear SVDD classifier [18]. This classifier uses bounding hyperspheres to approximate classes. The bounding hypersphere of a point set $\{\mathbf{x}_i | i = 1...n\}$ is characterized by its center $\mathbf{c}$ and radius r. These can be found by solving the quadratic programming problem

$$\begin{aligned} \underset{\mathbf{c},\, r\geq 0,\, \boldsymbol{\xi}\geq 0}{\arg\min}\ & \left(r^2 + \gamma \sum_i \xi_i \right) \\ \text{s.t. }\ & \|\mathbf{x}_i - \mathbf{c}\|^2 \leq r^2 + \xi_i, \quad i = 1, \ldots, n, \end{aligned} \tag{2}$$

or its dual

$$\begin{aligned} \underset{\boldsymbol{\alpha}}{\arg\min}\ & \left(\sum_{i,j} \alpha_i \alpha_j \langle \mathbf{x}_i, \mathbf{x}_j \rangle - \sum_i \alpha_i \|\mathbf{x}_i\|^2 \right) \\ \text{s.t. }\ & \sum_i \alpha_i = 1,\ \ \forall i\ \ 0 \leq \alpha_i \leq \gamma, \end{aligned} \tag{3}$$

where $\langle - \rangle$ represents the (possibly kernelized) inner product. The α_i are Lagrange multipliers and $\gamma \in [1/n, 1]$ is a ceiling parameter that can be set to a value less than one to reduce the influence of outliers. This problem is convex and its global optimum can be efficiently found by using a quadratic programming solver.

Negative samples can be used to improve the model by forcing them to lie outside of the bounding sphere. Suppose that we have n_1 face class samples enumerated by indices i, j, and n_2 background samples enumerated by l, m. The most compact bounding hypersphere that includes face samples and excludes the background samples can be found by solving the following quadratic programming problem

$$\begin{aligned} \underset{\mathbf{c},\, r\geq 0,\, \boldsymbol{\xi}\geq 0}{\arg\min}\ & \left(r^2 + \gamma_1 \sum_i \xi_i + \gamma_2 \sum_l \xi_l \right) \\ \text{s.t. }\ & \|\mathbf{x}_i - \mathbf{c}\|^2 \leq r^2 + \xi_i, \quad i = 1, \ldots, n_1 \\ & \|\mathbf{x}_l - \mathbf{c}\|^2 \geq r^2 - \xi_l, \quad l = 1, \ldots, n_2 \end{aligned} \tag{4}$$

or its dual

$$\begin{aligned} \underset{\boldsymbol{\alpha}}{\arg\min}\ & \sum_{i,j} \alpha_i\, \alpha_j\, \langle \mathbf{x}_i, \mathbf{x}_j \rangle + \sum_{l,m} \alpha_l\, \alpha_m\, \langle \mathbf{x}_l, \mathbf{x}_m \rangle \\ & - 2 \sum_{l,j} \alpha_l\, \alpha_j\, \langle \mathbf{x}_l, \mathbf{x}_j \rangle + (\sum_l \alpha_l \|\mathbf{x}_l\|^2 - \sum_i \alpha_i \|\mathbf{x}_i\|^2) \\ \text{s.t. }\ & \sum_i \alpha_i - \sum_l \alpha_l = 1,\ \forall i, j\ \ 0 \leq \alpha_i \leq \gamma_1,\ 0 \leq \alpha_l \leq \gamma_2. \end{aligned} \tag{5}$$

This problem is also convex, and it can be efficiently solved by using a quadratic programming solver. Given the optimal coefficients α_i, the center of the bounding sphere can be computed as

$$\mathbf{c} \ = \ \textstyle\sum_i \alpha_i\, \mathbf{x}_i - \sum_l \alpha_l\, \mathbf{x}_l\,. \tag{6}$$

The most of the α_i coefficients are zero and the samples corresponding to the nonzero coefficients are called the support vectors. It turns out that the majority of the support vectors come from the face class samples, and the number of the support vectors is much less than the number of support vectors returned by an SVM algorithm. Once we compute the center of the hypersphere, the radius can be found by using the constraints from (4).

During face detection, we find the distance from the feature vector of each local window to the center of the hypersphere and reject the sample as background if the distance is greater than radius. This is implemented efficiently since it only requires vector subtraction and computing the norm.

Nonlinear Hypersphere Approximation: The last stage of the cascade includes a kernelized hyperpshere classifier that makes the final decisions. The hypersphere model can be kernelized by replacing the inner products with kernel evaluations in (5). During detection, to evaluate the distances from incoming samples to the center of the bounding sphere, kernel evaluations against the support vectors are required. This makes kernelized SVDD classifier significantly expensive than its linear counterpart as the number of support vectors can be considerable. But since the kernelized hypersphere classifier operates on a few samples that pass all previous cascades, the overall speed is still good. In practice, kernelized SVDD classifiers are much faster than the analogous kernelized SVMs because they have far fewer support vectors since the support vectors come predominantly from the face training samples. As a result, kernel hypersphere classifier is better suited to use in efficient detection cascades than kernel SVM as demonstrated in [4].

B. Parts Detectors

For characterizing parts detector, we follow the procedure described in [7] with small changes. In our case, a part model for a face with n-parts is defined by a $(n+1)$-tuple $P_1, P_2, ..., P_n, b$ where P_i is a model for the i-th part and b is a real-valued bias. As in [7], each part model P_i is also defined by a 3-tuple (f_i, v_i, d_i) where f_i is a filter for the i-th part, v_i is a 2-dimensional vector specifying an anchor position for part i relative to the root position, and d_i is a 4-dimensional vector that specifies the coefficients of a quadratic function defining the deformation cost. The final score of a part detector at position $l_i = (x_i, y_i)$ within any image feature space H is computed as

$$\text{score}(P_1, ..., P_n) = \sum_{i=1}^{n} f_i^\top \Phi(H, P_i) - \sum_{i=1}^{n} d_i \Phi(dx_i, dy_i) + b, \tag{7}$$

where (dx_i, dy_i) gives the displacement of the i-th part relative to its anchor position and $\Phi(dx_i, dy_i) = (dx_i, dy_i, dx_i^2, dy_i^2)$ are deformation features, and $\Phi(H, P_i)$ represents image feature for the i-th part. Concatenating these parameters into a single vector $\boldsymbol{\beta}$, the score can be written as $\boldsymbol{\beta}^\top \Phi(H, z)$.

We use root filter (the normal of the separating hyperplane returned by the SVM classifier during roots detector training) as in [7] to initialize the part filters. We set the number of parts to be 3 (two parts for the eyes and one part for the mouth) and we manually place the parts into the regions where the eyes and mouth occur in a frontal face view as shown in Fig. 2. The part filters are initialized by interpolating the root filter to twice the spatial resolution. The deformation parameters are initialized to $d_i = (0, 0, 0.05, 0.05)$. By using initial model, we extract features $\Phi(H, z)$ belonging

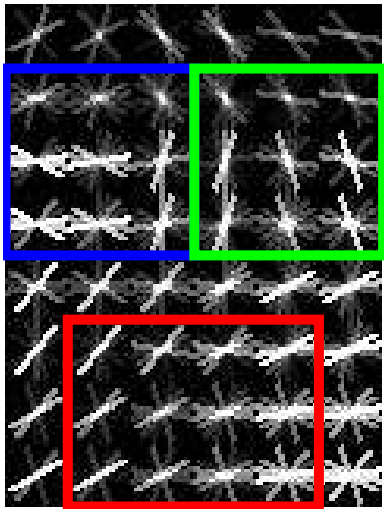

Fig. 2. Initialization of part filters from the root filter. We use 3 parts to detect eyes and mouth. Colored rectangles show the initial parts filters.

to the positive and negative samples and iteratively learn the new model parameters by solving the following SVM-like quadratic optimization problem

$$\begin{aligned} \underset{\boldsymbol{\beta}, \boldsymbol{\xi} \geq 0}{\arg\min} \; & \frac{1}{2} \boldsymbol{\beta}^\top \boldsymbol{\beta} + C \sum_i \xi_i \\ \text{s.t.} \; & \boldsymbol{\beta}^\top \Phi(H, z_i) + b \geq 1 - \xi_i, \;\; \forall i \in I_+, \\ & \boldsymbol{\beta}^\top \Phi(H, z_i) + b \leq -1 + \xi_i, \;\; \forall i \in I_-, \\ & \beta_k \leq 0, \forall k \in D, \end{aligned} \tag{8}$$

where D is the set of indices corresponding to the quadratic deformation terms, dx^2, dy^2. This is different than the classical SVM formulation in the sense that we force the filter coefficients corresponding to the quadratic terms to be negative so that they will be always subtracted during computation of parts scores. As in linear SVM classifier, the positive examples will be separated from the negative examples by a margin $2/||\boldsymbol{\beta}||$.

To solve (8) we adapted the optimized cutting plane (OCA) algorithm [8] which has been proved efficient for solving large-scale quadratic problems emerging in the SVM training. The OCA algorithm incrementally constructs a cutting plane model of the problem using a novel strategy to efficiently approximate the objective function around its optimum. The cutting plane model is used to find a search direction in the parameter space and a new iterate is obtained by an exact line-search procedure. In contrast to the standard SVM training, the problem (8) contains additional constraints preventing a subset of parameters to be positive. This modification requires slight changes to the original OCA algorithm. In particular, the new constraints must be incorporated to the exact line-search procedure and the inner quadratic solver. Our implementation of the solver can be downloaded from *http://cmp.felk.cvut.cz/∼xfrancv/ocas/html/index.html*.

III. EXPERIMENTS

Our face detector is trained by using 21K face images collected from web and various face recognition databases and 100K negative examples. We used LBP+HOG features as described before. We used only positive face samples to learn linear hypersphere model (optimization problem given in (3)) and used both negative and positive samples to learn kernelized hypersphere model (optimization problem given in (5)). We tested our detector[1] on two face detection datasets and compared our results with those of OpenCV Viola-Jones cascade detector [19], boosted frontal face detector of Kalal et al. [11], our previous cascade detector [4] that includes a single root detector, and part-model based detector of Zhu and Ramanan [23]. PASCAL VOC metric is used to assess the detection performance. In this metric, detections are considered as true or false based on the overlap with ground truth bounding boxes. The overlap between the bounding boxes, R returned by the classifiers, and the ground truth box Q, is computed as $\frac{\text{area}|Q \cap R|}{\text{area}|Q \cup R|}$. Bounding boxes with an overlap of 45% (we used 45% rather than 50% to compensate for the differences between bounding box annotation choices between different people and detector outputs) or greater are considered as true positives. Then we report average precision (AP) for the whole Precision-Recall curve.

A. Results on Faces in the Wild Dataset

We have used 13127 images including frontal faces from this dataset. It should be noted that the majority of the faces appear in the middle, and they are scaled to fit to the same resolution. This limits its value as test set for multi-scale face detectors. In our proposed method, search window size for root detectors is 36×30, and part detectors are applied to windows with size 72×60 (twice the root detector window size). Part-based detector of Zhu&Ramanan [23] returns only face parts and face region is estimated as the tightest bounding box including all face parts. This method works with high-resolution images where the faces are larger than 80×80 pixels. The images in the Faces in the Wild dataset are typically high-resolution images, where the faces in the images are around 90×100 pixels. Thus, part-based methods are successfully applied. The AP scores are given in Table I. The best detection accuracy is obtained by part-based detector of Zhu&Ramanan [23]. Our proposed method comes the second followed by the our previous method [4] using only one root detector (without parts detectors). Cascade detector of Viola&Jones [19] is the worst performing method. In general, part-based detectors yield more successful detection accuracies for this database.

TABLE I

AVERAGE PRECISION (AREA UNDER CURVE) SCORES (%) ON THE FACES IN THE WILD DATABASE.

Methods	Average Precision
Proposed Method	94.39
Viola&Jones [19]	80.23
Kalal et al. [11]	87.89
Cevikalp&Triggs [4]	94.12
Zhu&Ramanan [23]	96.90

B. Results on Extended ESOGU Face Detection Database

We also tested detectors on extended ESOGU (ESkisehir OsmanGazi University) frontal face database[2]. The original database contained 285 higher-resolution color images including faces that appear at a wide range of image positions and scales, and also complex backgrounds, occlusions and illumination variations. We extended the database by adding 382 new images. Thus, the database now includes 667 images that contains 2042 annotated frontal faces. The AP results of tested detectors are given in Table II, and corresponding Precision-Recall curves are plotted in Fig. 3. The best performing method of Zhu&Ramanan [23] on Faces in the Wild database gives the worst accuracy for this database since there are many faces smaller than 80×80 pixels in this database. The best accuracy is achieved by our previous cascade detector [4] followed by the proposed method. For many faces, the sizes of the images are too small to search for the parts, thus we upsample the face region in such cases. But, returned parts mostly fail for upsampled low-resolution faces. The face detector of Kalal et al. [11] comes the third best performing method. We also compared the results to a commercial system Google Picasa[3]. To this end, we visually counted the number of faces returned by the people tagging tool of this system. We considered all returned face images as true positives although some faces do not satisfy PASCAL VOC overlapping criterion. We also ignored all false positives coming from the background. In this case, the recall rate is computed as 91.52% and it is plotted in Fig. 3.

TABLE II

AVERAGE PRECISION (AREA UNDER CURVE) SCORES (%) ON THE EXTENDED ESOGU FACE DETECTION DATABASE.

Methods	Average Precision
Proposed Method	83.86
Viola&Jones [19]	73.61
Kalal et al. [11]	79.67
Cevikalp&Triggs [4]	87.05
Zhu&Ramanan [23]	50.55

C. Discussion

Our tests on challenging data sets show that the current face detectors mostly tolerate to small rotations and illumination changes, but fail to detect faces when the rotations and illumination changes are severe. They also typically cannot detect faces smaller than 30×30 pixels. Part-based models work well if the face images are large and eyes and mouth are clearly visible within the face region. If this condition is

[1] Available at *http://www2.ogu.edu.tr/∼mlcv/softwarecvpr2012.html*

[2] Available at *http://mlcvdb.ogu.edu.tr/facedetection.html*

[3] Available at *http://picasa.google.com*

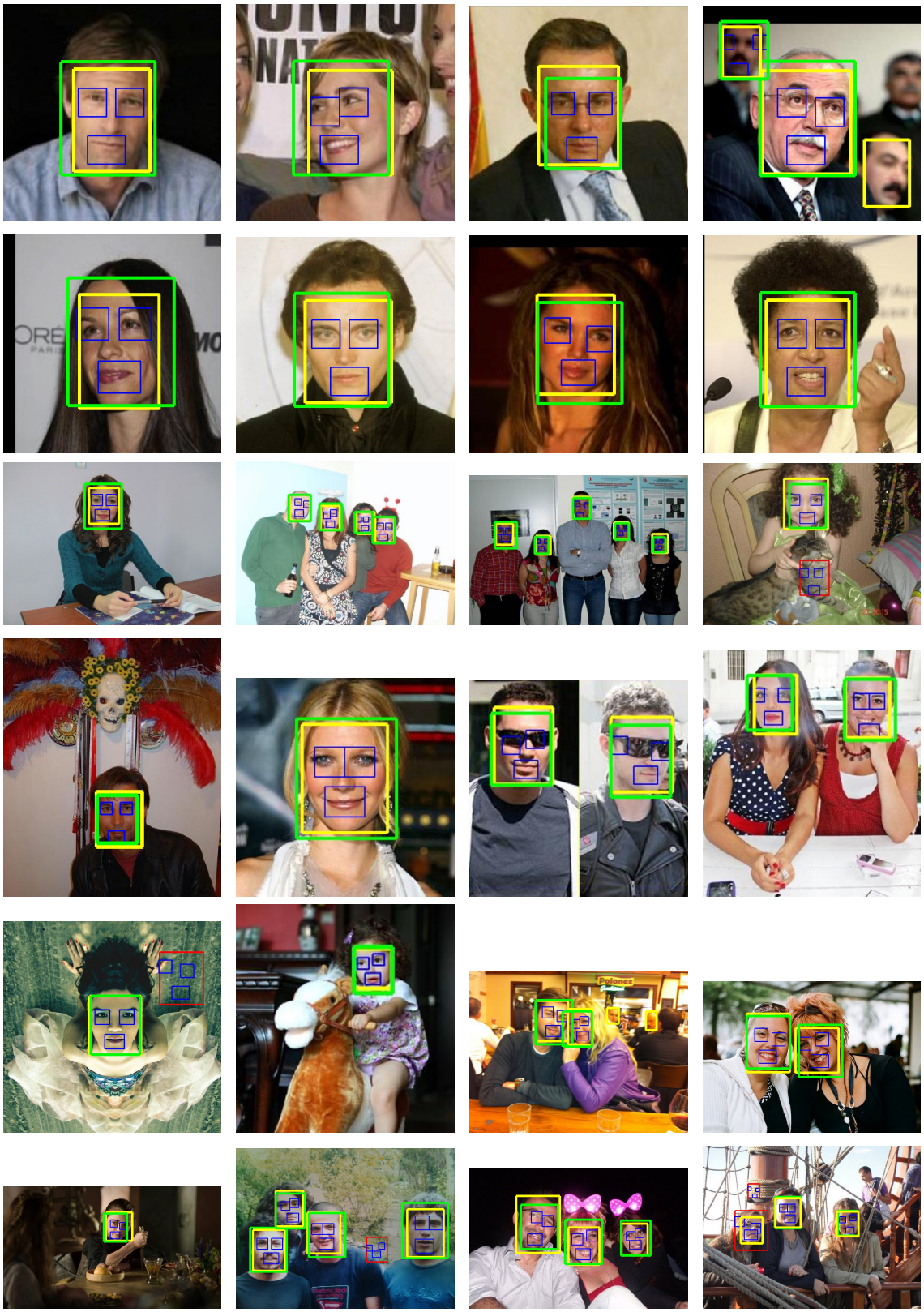

Fig. 4. Some examples of the output of our cascade detector on images from the Faces in the Wild dataset (top two rows) and ESOGU face dataset (the remaining rows). Yellow rectangles show true human annotations, green rectangles show the correct detections based on PASCAL VOC criterion, and red rectangles show the false positives. Most of the faces are correctly detected, but there are a few missed detections and false positives. Part detectors typically locate the eyes and mouth correctly if the face image region is large, but mostly fail for the small faces. Part detectors also fail to locate eyes if the person wears sunglasses or if the returned face region is much larger or smaller than the actual face area (e.g., the last example on the bottom right).

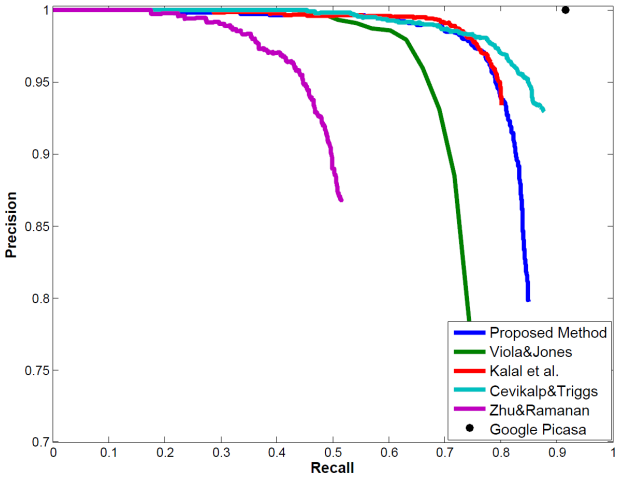

Fig. 3. Presicion-Recall curves for the extended ESOGU face detection database.

not met, it is better to rely only on the root detectors (for low-resolution images, simple techniques such as thresh-holding combined with prior information on the location of fiducial points can suffice to locate face fiducial points).

Results also show that there is not a significant gap between the performances of the best face detectors introduced by academic community and commercial face detectors. The results are encouraging, and both face and landmark detectors can be integrated into face recognition systems without hesitation (most of the face recognition methods proposed in the literature rely on manual face alignment and there are only a few studies that implement a fully-automatic face recognition system). Also results on current face recognition databases are mostly saturated, and it is necessary to introduce more challenging face recognition databases. We would also like to point out a common mistake of face recognition methods. In most methods, the face images are down-sampled to small sizes such as 40×40 pixels. However, this is a mistake in our opinion since most of the details of the face region and parts are lost in such low-resolutions (face recognition methods work well even for this small sizes since the current face recognition databases are too easy in the sense that they are captured under controlled environments).

IV. CONCLUSION

We introduced a novel face detector to find both faces and face-parts by using the roots and parts detectors. Roots detectors use a cascade of binary and one-class classifiers. As a binary classifier we use linear SVM, and we use linear hyperplane and linear/kernelized hypersphere models for approximation of face classes in one-class classifiers. Parts detectors are traind by using a variation of SVM classifier that enforces additional constraints on the weights of the normal vector. Our proposed detector outperforms the most of the succesfull face detection algorithms and gives the second best result on all tested challenging databases.

Acknowledgments: This work was funded in part by the Scientific and Technological Research Council of Turkey (TUBİTAK) under Grant number EEEAG-109E279 and the Young Scientists Award Programme (TÜBA-GEBİP/2011-12) of the Turkish Academy of Sciences. The last author was supported by EC project FP7-ICT247525 HUMAVIPS.

REFERENCES

[1] T. Ahonen, A. Hadid, and M. Pietikainen. Face description with local binary patterns: Application to face recognition. *IEEE Transactions on PAMI*, 28(12):2037–2041, 2006.
[2] K. P. Bennett and E. J. Bredensteiner. Duality and geometry in svm classifiers. In *ICML*, 2000.
[3] H. Cevikalp and B. Triggs. Nearest hyperdisk methods for high-dimensional classification. In *International Conference on Machine Learning*, 2008.
[4] H. Cevikalp and B. Triggs. Efficient object detection using cascades of nearest convex model classifiers. In *CVPR*, 2012.
[5] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *CVPR*, 2005.
[6] M. Dantone, J. Gall, G. Fanelli, and L. V. Gool. Real-time facial feature detection using conditional regression forests. In *CVPR*, 2012.
[7] P. Felzenszwalb, R. B. Girshick, D. McAllester, and D. Ramanan. Object detection with discriminatively trained part based models. *IEEE Transactions on PAMI*, 32(9), Sept. 2010.
[8] V. Franc and S. Sonnenburg. Optimized cutting plane algorithm for large-scale risk minimization. *Journal of Machine Learning Research*, 10:2157–2232, October 2009.
[9] C. Garcia and M. Delakis. A convolutional face finder: a neural architecture for fast and robust face detection. *IEEE Transactions on PAMI*, 26(11):1408–1423, 2004.
[10] H. Jin, Q. Liu, and H. Lu. Face detection using one-class-based support vectors. In *International Conference on Automatic Face and Gesture Recognition*, 2004.
[11] Z. Kalal, J. Matas, and K. Mikolajczyk. Weighted sampling for large-scale boosting. In *BMVC*, 2008.
[12] W. Kienzle, G. Bakir, M. Franz, and B. Scholkopf. Face detection - efficient and rank deficient. In *NIPS*, 2005.
[13] K. Levi and Y. Weiss. Learning object detection from a small number of examples: the importance of good features. In *CVPR*, 2004.
[14] C. Papageorgiou and T. Poggio. A trainable system for object detection. *IJCV*, 38:15–33, 2000.
[15] H. A. Rowley, S. Baluja, and T. Kanade. Neural network-based face detection. *IEEE Transactions on PAMI*, 20:23–38, 1998.
[16] L. Shams and J. Speslstra. Learning Gabor-based features for face detection. In *World Congress in Neural Networks*, 1996.
[17] X. Tan and B. Triggs. Enhanced local texture feature sets for face recognition under difficult lighting conditions. *IEEE Transactions on Image Processing*, 19:1635–1650, 2010.
[18] D. M. J. Tax and R. P. W. Duin. Support vector data description. *Machine Learning*, 54:45–66, 2004.
[19] P. Viola and M. J. Jones. Robust real-time face detection. *IJCV*, 57(2):137–154, 2004.
[20] C. A. Waring and X. Liu. Face detection using spectral histograms and svms. *IEEE Transactions on Systems, Man, and Cybernetcis-Part B: Cybernetics*, 35:467–476, 2005.
[21] M. Yang, D. Kriegman, and N. Ahuja. Detecting faces in images: A survey. *IEEE Transactions on PAMI*, 24(1):34–58, 2002.
[22] L. Zhu, Y. Chen, A. Yulie, and W. Freeman. Latent hieararchical structural learning for object detection. In *CVPR*, 2010.
[23] X. Zhu and D. Ramanan. Face detection, pose estimation, and landmark localization in the wild. In *CVPR*, 2012.