

# Fast Face Detection with a Boosted CCCD Classifier

Diego A. Socolinsky\*, Joshua D. Neuheisel\*, Carey E. Priebe<sup>†</sup>, Jason DeVinney<sup>‡</sup>, David Marchette<sup>‡</sup>

Equinox Corporation\*  
207 East Redwood Street  
Baltimore, MD 21202

Department of Mathematical Sciences<sup>†</sup>  
The Johns Hopkins University  
Baltimore, MD 21218

Naval Surface Warfare Center<sup>‡</sup>  
B10  
Dahlgren, VA 22448

## Abstract

We introduce a fast object detection algorithm based on the Class-Cover Catch Digraph (CCCD) classifier. When applied to face detection, our algorithm exhibits good performance with speeds close to those of the fastest reported techniques. The main technical innovations in our method include a boosted tree-like CCCD classifier with a maximum rejection bias, and the use of a cross-correlation metric for fast similarity computation.

## 1 Introduction

Detection of faces in everyday images is a necessary first step to any facial recognition system, and as such constitutes a problem of the most current importance. In order for a detector to be useful as a front-end to most recognition systems, it must be capable of operating at high frame-rates on images of sufficient resolution to perform recognition on the detected faces. Most face detectors capable of operating at high frame rates do so by taking advantage of attentional cues, such as color or motion [11, 6, 8, 15, 5] to restrict application of a classifier to small regions of the input image. While this can be an effective means of detecting faces in real time, it is still of interest to construct a face detector capable of operating on static grayscale imagery at high speeds. Until recently [17, 18] such detectors were all but non-existent.

This paper introduces a new method for finding human faces in static grayscale images at high speeds without resorting to specialized hardware. Our method is based on previous work by the authors on a classifier designed for high dimensional problems, named the Class-Cover Catch Digraph (CCCD) classifier. This is a generic classification methodology, stemming from the simple nearest-prototype family of classifiers and optimized for problems where we have sparse training data with respect to the dimensionality of the feature space [1, 7]. The rate at which a face detector can discard non-face regions of an image is to a large extent the dominant factor determining its performance, as

non-faces are many times more abundant than faces [2]. In order to achieve high correct classification rates on the non-face class the CCCD is structured as a degenerate decision tree, where successive levels boost rejection rates for previous ones while maintaining a bound on the error incurred on the face class. This structure also allows the detector built upon the classifier to operate at high speeds, since each boosted tree layer rules out more and more of the image as a potential face, and hence limits the need for further processing to those image regions that appear most promising for containing faces. An additional speed increase is achieved by computing Euclidean distances by means of fast normalized cross-correlation in the frequency domain for a portion of the algorithm.

The contributions below are organized as follows. First we describe the training data used for faces and non-faces, as well as the simple normalization process applied to it. While more involved normalizations would probably increase detection performance, they would also incur a computational penalty, so we stick to our simple but fast normalization. The next section briefly describes the standard CCCD classifier, but the reader may wish to consult the references for a more complete account. A boosted tree version of the CCCD classifier is introduced in Section 4, where we show how successive filtering of the training data along with a procedure to bias the classification outcome of the standard CCCD classifier can be used effectively as a boosting technique. Section 5 shows how the boosted tree of the previous section can be evaluated efficiently on an image for fast detection. Details about merging multiple detections are taken up in Section 6. Finally, we present some experimental results showing the performance of our proposed algorithm.

## 2 Training Data

The classifier described below was trained for face detection using large sets of face and non-face data. The face training set was constructed using face images from the FERET database and the Equinox HID database, totaling almost



Figure 1: Sample face and non-face training data.

2000 examples. Each image was normalized so that the manually extracted eye locations would appear at fixed coordinates, cropped and re-scaled to  $21 \times 21$  pixels. Additionally, in order to provide some measure of illumination compensation, the mean image grayvalue was subtracted from each image, and the resulting data normalized to have unit norm. The same normalization process was applied to all non-face image examples, which were collected from several hundred images downloaded from the world-wide-web and manually verified to contain no faces. A large number of  $21 \times 21$  pixel sub-windows (over a million) were extracted from these original images at a number of scales. A few tens of thousands were used to train the first classifier sub-stage, and the rest were used for the boosting process.

### 3 The CCCD Classifier

This section provides a brief introduction to the Class-Cover Catch Digraph (CCCD) classifier. A more detailed description and analysis can be found in [1]. In particular, this works uses a derivative of the random-walk CCCD classifier introduced in [7].

The CCCD classifier is a nearest-prototype classifier with respect to a non-linear dissimilarity function. For simplicity, consider as training data two sets of class-conditional  $\mathbb{R}^n$ -valued observations  $\mathcal{X}_0$  and  $\mathcal{X}_1$ , and a dissimilarity measure  $\rho : \mathbb{R}^n \times \mathbb{R}^n \rightarrow \mathbb{R}_+$ , satisfying  $0 = \rho(x, x) < \rho(x, y) < \infty$ , for  $x \neq y \in \mathbb{R}^n$ . The goal of classifier design is to construct a function  $g_{\mathcal{X}_0, \mathcal{X}_1} : \mathbb{R}^n \rightarrow \{0, 1\}$  such that for a given unlabeled observation  $x \in \mathbb{R}^n$  with unknown class label in  $y \in \{0, 1\}$ , the probability of misclassification  $P[g_{\mathcal{X}_0, \mathcal{X}_1}(x) \neq y]$  is close to Bayes optimal [3, 14].

For a set of prototypes  $C_i = \{c_{i,1}, \dots, c_{i,k}\} \subset \mathcal{X}_i$  and  $R_i = \{r_1, \dots, r_k\} \subset \mathbb{R}_+$ ,  $i = 0, 1$ , the CCCD *cover-dissimilarity measure* is defined by

$$d(x, C_i) = \min_k \frac{\rho(x, c_{i,k})}{r_k}. \quad (1)$$

Given sets  $C_i$  and  $R_i$  as above, the CCCD classifier is defined in terms of the cover-dissimilarity measure (1) by

$$g_{\mathcal{X}_0, \mathcal{X}_1}(x) = \arg \min_i d(x, C_i). \quad (2)$$

The choice of prototypes  $C_i$  and scaling factors  $R_i$ , determines the classifier map. Below, we give a short account of the method for choosing these parameters. A more thorough account can be found in [1, 7], along with performance analyses for applications other than face detection.

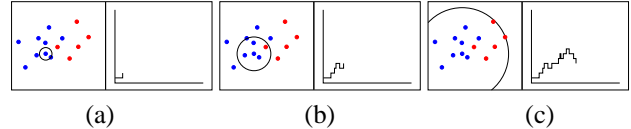


Figure 2: Example of random walk construction for a two-dimensional problem.

For each point  $x_{i,j} \in \mathcal{X}_i$ , we consider the random walk defined as follows

$$R_{x_{i,j}}(r) = |\{x \in \mathcal{X}_i : \rho(x_{i,j}, x) < r, \} - \{x \in \mathcal{X}_{(1-i)} : \rho(x_{i,j}, x) < r, \}|, \quad (3)$$

for  $r \in \mathbb{R}_+$ ,  $i = 0, 1$ . A large value of  $R_{x_{i,j}}$  indicates a high local density of same-class points around  $x_{i,j}$ , relative to the local density of other-class points (see [7] for the case of unequal training priors). In fact, the value of the random walk at any given  $r \in \mathbb{R}_+$  can be taken as a measure of relative deviation between the local densities of the training data. A Kolmogorov-Smirnov type test is applied in [7] to obtain a distinguished choice of  $r$  for each training observation, given by

$$r_{x_{i,j}}^* = \arg \max_r R_{x_{i,j}}(r) - P(r), \quad (4)$$

where  $P(r)$  is an increasing penalty function that biases the choice toward smaller values of  $r$ . This is done in order to encourage more local estimation of the classifier parameters, and is achieved in practice by the use of a linear penalty function.

Once a distinguished scaling factor  $r_{x_{i,j}}^*$  has been chosen for each training observation, it remains to find the choice of class-conditional prototypes  $C_i$  that will fully determine the CCCD classifier. The procedure in the standard CCCD classifier is somewhat different than the one used in this paper (described in detail below), but we include a summary of it for completeness. Ideally, the choice of prototypes would be that which maximizes classifier performance. However, the combinatorial explosion involved in checking all possible prototype sets precludes a direct approach. The choice

of prototypes for each class proceeds in a greedy fashion, using a surrogate criterion for the classifier performance

$$T_{x_{i,j}} = R_{x_{i,j}}(r_{x_{i,j}}^*) - P(r_{x_{i,j}}^*). \quad (5)$$

For a given class, the first prototype  $c_{i,1}$  is chosen to be that with the highest value of  $T$ . All training observations  $x$  for which  $\rho(x, c_{i,1}) < r_{c_{i,1}}^*$  are then deleted from the training set, all scaling factors  $r^*$  are recomputed for the remaining training observations, and the next prototype is chosen using the surrogate criterion  $T$  as before. This process continues until all but a predetermined portion of the class- $i$  training data has been deleted.

## 4 A Boosted CCCD Classifier

Although some degree of boosting is inherent to the CCCD classifier (by means of censoring the training data in the greedy prototype selection process), increased performance can be achieved by more explicit boosting during training. Additionally, it is advantageous to exploit the fact that in natural scenes, the relative likelihood of observing background (class 1) at any given image location is several orders of magnitude larger than that of observing a face (class 0). Therefore, classifier design should be geared toward rejecting the background class, both for accuracy and performance reasons [2]. In our case, we will achieve this by boosting only on the background (non-face) class and structuring the classifier as a one-sided decision tree.

Note that by virtue of the nature of the surrogate criterion (5), the CCCD classifier (2) for a set of prototypes  $C_i = \{c_{i,1}, \dots, c_{i,k}\} \subset \mathcal{X}_i$  and scaling factors  $R_i = \{r_1, \dots, r_k\} \subset \mathbb{R}_+$  is dominated by the influence of the first few elements of each set. In particular, the first prototype and scaling factor are the most important in determining the classification map. Also, note that we can easily bias the classification performance of any CCCD classifier in favor of lower type-I or type-II error by modifying the scaling factors as

$$\tilde{R}_0 = t R_0, \quad \tilde{R}_1 = t^{-1} R_1, \quad (6)$$

for  $0 < t < \infty$ . Values of  $t$  in  $(0, 1)$  favor lower class-1 error at the expense of higher error rate on class-0, and vice-versa. These two observations lead to the boosted CCCD tree introduced below.

Training of a boosted CCCD tree is separated into stages and sub-stages. In what follows, we associate prototypes and their corresponding scaling factors, and we refer to them simply as prototypes. The first prototype in each class is selected by the same procedure as in Section 3 (if more class-0 prototypes are desired, they are all chosen in this step, but we describe the process for a 1-prototype stage, for simplicity). At this point, we have a simple CCCD

classifier, with one prototype per class. Using the biasing procedure of Equation (6) and a test set of class-0 observations, we find the lowest value of  $t$  (or a suitable approximation) that yields an empirical error rate below a predetermined fixed tolerance. The two prototypes along with the resulting scaling factors constitute the first sub-stage of the CCCD tree stage. In order to compute the second sub-stage, we apply the first sub-stage classifier (see Algorithm 2) to a set of class-1 data and collect the misclassified observations, which become the class-1 training observations for the second sub-stage. Using once again the procedure in Section (3), a single class-1 prototype is selected, and through the bias procedure in (6), the scaling factors for the single (fixed) class-0 prototype and the newly chosen class-1 prototype are computed. This process is repeated as many times as necessary to obtain the desired number of sub-stages.

The number of sub-stages within a given stage is empirically determined; we continue to add sub-stages to a stage as long as the correct classification rate on the background class increases by a significant (predetermined) percentage over that of the previous sub-stage. At that point, we start training an entirely new classifier stage, using the original class-0 training data, and the class-1 training data that is misclassified by all previous classifier stages. While speed considerations (see Section 5) dictate that the first stage of the tree have a single class-0 prototype, subsequent stages are allowed to have multiple face prototypes. In fact, it is natural to allow later stages to use more face prototypes than earlier ones, thus allowing the classifier to more closely model the support of the face distribution. Algorithm 1 shows the steps in the boosted training algorithm. Here, the indexes  $i$  and  $j$  correspond to stages and sub-stages, respectively. Each stage  $i$  is defined by its  $f_i$  face prototypes  $C_0^i = \{c_{0,1}^i, \dots, c_{0,f_i}^i\}$  and scaling factors  $R_0^i$ , as well as the  $n_i$  non-face prototypes and scaling factors  $C_1^i = \{c_{1,1}^i, \dots, c_{1,n_i}^i\}$  and  $R_1^i$  corresponding to each sub-stage. A large (on the order of a million or more) set of non-face observations  $\mathcal{T}_1^0$  is used for the boosting process, and a separate set  $\mathcal{T}_1^1$  of face samples is used to evaluate the empirical classifier performance on the face class.

The main emphasis of our work lies in lowering the classifier error rate and speeding up its application at time of detection. Training time is not a major factor, since training occurs off-line and once completed does not need to occur again; that is once a face detector has been trained, it can be used indefinitely without modification. However, if training times are prohibitively long, then it is not possible to obtain the desired detector. In our case, the large number of training observations used, especially for the non-face class, would make standard CCCD training as in [1] and [7] all but impossible, since the algorithm there requires the computation of all distances between training observations. In or-

der to avoid this, we apply the following stochastic search strategy. At each sub-stage of training, random subsamples from the class-0 and class-1 training data are drawn, consisting of approximately 200 observations each. These subsets are used to train the sub-stage as above, and the empirical performance of the resulting stage on class-1 data is evaluated on the class-1 data not used for training (the complement of the random sample). Recall that the performance on class-0 data is enforced explicitly, so it is not necessary to evaluate it. This process is repeated multiple times for new random samples of the training data, and the sub-stage yielding highest empirical performance is in the final classifier. Experimentally, we have observed that if the random training subsets are not too small, this procedure yields a classifier whose performance is indistinguishable from that of one trained on the full set of training data, but requiring only a small fraction of the computation time. We normally use on the order of 100 random iterations of the above procedure for each sub-stage of the classifier tree. Figure 3 shows classifier performance on each class as a function of the total sub-stage count, for a 42 sub-stage tree. Note how the performance on the face class decreases as the number of sub-stages increases<sup>1</sup>, while the error rate on the background class decreases (starting with almost 93% correct classification using a single sub-stage). A straight sum of the error rates is not a good criterion of performance, however, as the priors are severely skewed (toward the background class). Hence, even though it would appear that the optimal classifier in this case has 5 sub-stages, the full tree indeed has better performance.

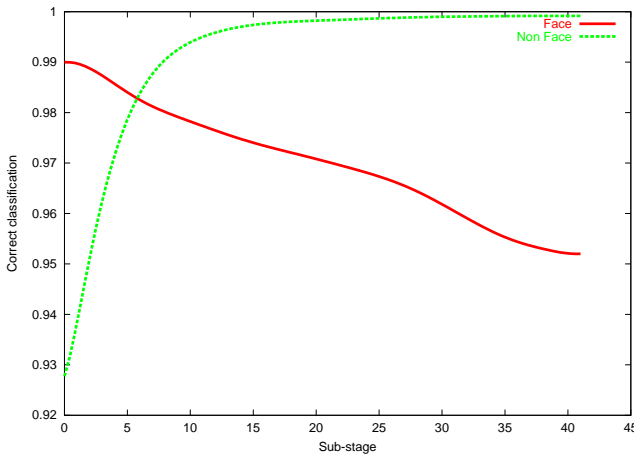


Figure 3: Boosted tree performance as a function of total sub-stage count, for a 42 sub-stage classifier.

<sup>1</sup>The error rate on the face class is bounded above by the number of sub-stages times the error bound on each individual sub-stage.

---

#### Algorithm 1: Boosted training algorithm.

---

**Data** : *i.i.d* sets of class-0 and class-1 data  $\mathcal{T}_0$  and  $\mathcal{T}_1^0$ , respectively. Required number of stages  $\beta$ . Threshold on the incremental classification improvement on class-1,  $\alpha$ .

Let  $i = j = k = 1$ ,  $\mathcal{X}_1^0 = \mathcal{X}_1$ ;

**while**  $i \leq \beta$  **do**

Select  $C_0^i = \{c_{0,1}^i, \dots, c_{0,f_i}^i\}$  and  $R_0^i$  as in Section 3;

**repeat**

Select  $c_{1,j}^i$  and  $r_{1,j}^i$  as in Section 3, using  $\mathcal{X}_1^{k-1}$  as training data;

Adjust the scaling factors as in Equation 6 to enforce the required empirical performance bound on the face class;

Let  $\mathcal{T}_1^k \subseteq \mathcal{T}_1^0$  be the class-1 observations incorrectly classified by the current classifier, and  $\mathcal{X}_1^k$  be the misclassified class-1 training observations;

**until**  $|\mathcal{T}_1^k|/|\mathcal{T}_1^{k-1}| \geq \alpha$ ;

**end**

---

## 5 Fast Detection

The general architecture of our detector is similar to that of many others proposed in the literature [2, 17, 4, 16]. Essentially, every sub-window of a fixed size in the original image is classified as face or background at its original scale. Then the image is zoomed down by a predefined factor, and the classification process is repeated. This section explains how this potentially exhaustive process can in fact be performed swiftly and efficiently enough for near real-time operation on off-the-shelf hardware.

When trained as in Section 4, a CCCD classifier can achieve very good performance distinguishing faces from

---

#### Algorithm 2: Boosted tree classification.

---

**Data** : Number of stages  $\beta$ . Number of face prototypes per stage,  $f_i$ , and number of sub-stages per stage,  $n_i$ , for  $i = 1, \dots, \beta$

Let  $i = 1$ ;

**while**  $i \leq \beta$  **do**

Let  $m_0 = \min_{k=1}^{f_i} \rho(x, c_{0,k}^i)$ ;

**for**  $j = 1, \dots, n_i$  **do**

**if**  $\rho(x, c_{1,j}^i) < m_0$  **then**

Classify as non-face and exit;

**end**

**end**

**end**

Classify as face;

---

non-faces. This classifier can be immediately applied to an image by extracting every possible subwindow of the size used for training and labeling as face or non-face according to the classifier output. Unfortunately, this method yields an exceedingly slow face detector. We can reach near real-time performance by taking advantage of two architectural speed-ups.

Firstly, we can structure the detection process hierarchically, as a decision tree. This method has been successfully used for face detection in [2, 17, 18] and others. Note that the classifier developed in Section 4 has the structure of a degenerate decision tree, where a sample is labeled as a face if-and-only-if it is not determined to be background at any stage of the decision process. Therefore, only those samples that have not been labeled as background by earlier classifier stages need to be considered by later ones. Consequently, much computing time may be saved if a large number of image subwindows can be discarded early on in the classification process. Figure 4(a) shows an example of the relative number of classifier sub-stages applied to each image pixel, with brighter regions indicating more sub-stages. We see that most of the image is discarded using a small portion of the entire decision tree, and only those areas where faces are located make use of all classifier sub-stages, thus only few pixels necessitate heavy processing. In fact, the average number of sub-stages needed to properly classify images from our non-face test set is under 1.5.

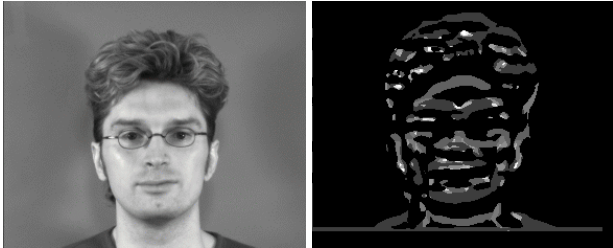


Figure 4: Example image showing the relative number of classifier stages applied at each pixel (in logarithmic scale). Each pixel in the bottom image represents the upper-left corner of a detection box, and is thusly shifted up and to the left with respect to the original image.

While the hierarchical approach outlined above is very effective at lowering the computational burden exacted by subsequent stages of the detection pipeline, it does nothing to improve the situation for the very first sub-stage. In that case, all image pixels must be considered, since we do not have any *a priori* information as to where faces may be located in the image. This information is often present in systems relying on color, motion or other cues for initialization, and can be incorporated into the current system, if available (see [10] and the references therein). It is possible

to exhaustively consider all image sub-windows in a naïve fashion, but this is much too slow for near real-time operation. We show instead how to process the first sub-stage in the Fourier domain, thus achieving much faster run times. A similar approach was proposed for a neural network based system in [13].

We use the  $L^2$  norm for  $\rho$  in Equation (1). In order to compute this efficiently, recall that for two grayscale image patches  $I$  and  $K$ , the *normalized cross correlation* is defined as:

$$NCC(I, K) = \frac{(I - \bar{I}) * (K - \bar{K})}{\|I - \bar{I}\| \|K - \bar{K}\|}, \quad (7)$$

where  $\bar{I}$  and  $\bar{K}$  denote the mean values of  $I$  and  $K$  respectively, and  $*$  denotes the convolution operator. If we denote the  $L^2$  distance between  $I$  and  $K$  as vectors by  $L^2(I, K)$ , then we see that if  $\bar{K} = 0$  and  $\|K\| = 1$ , then

$$L^2\left(\frac{I - \bar{I}}{\|I - \bar{I}\|}, K\right) = \sqrt{2} \sqrt{1 - NCC(I, K)}, \quad (8)$$

thus the  $L^2$  norm used by the CCCD classifier can be computed in terms of the normalized cross correlation.

In turn, the NCC can be computed using one Fourier transform plus a fast table computation introduced in [9], and replicated in a different context in [17, 18]. It is well known that convolution in the image domain is equivalent to pointwise multiplication in the Fourier domain, and we use this fact to compute the convolution component in Equation (7). While it is also possible to compute the means and norms in Equation (7) using Fourier methods, it turns out to be faster to do it using the tabular procedure described in [9], as follows. Indexing the image  $I$  by rows and columns  $0 \leq i \leq N$  and  $0 \leq j \leq N$ , respectively, we have the tables of cumulative sums

$$S_{i,j} = I_{i,j} + S_{i-1,j} + S_{i,j-1} - S_{i-1,j-1} \quad (9)$$

$$S_{i,j}^2 = I_{i,j}^2 + S_{i-1,j}^2 + S_{i,j-1}^2 - S_{i-1,j-1}^2, \quad (10)$$

where  $S_{i,j} = 0$  if  $i < 0$  or  $j < 0$ . Now we note that the mean value of the sub-image of  $I$  with upper-left-hand corner at coordinates  $(i, j)$  width and height  $w$  and  $h$ , respectively, can be read off the table  $S$  as

$$\frac{1}{wh} [S_{i+N-1,j+M-1} - S_{i-1,j+M-1} - S_{i+N-1,j-1} + S_{i-1,j-1}] \quad (11)$$

The norms in Equation (7) can be similarly computed from the corresponding table  $S^2$  as

$$S_{i+N-1,j+M-1}^2 - S_{i-1,j+M-1}^2 - S_{i+N-1,j-1}^2 + S_{i-1,j-1}^2 \quad (12)$$

For the case of the boosted CCCD classifier face detector, the image  $I$  corresponds to the image in which we wish to detect faces, and  $K$  represents the first (face or non-face) prototype obtained through CCCD training. Note that in our case  $K$  always has zero mean and unit norm, so the only table computations to be performed are those needed to locally normalize the input image  $I$ . Also note that this procedure needs to be repeated for each scale of the face detection process. However, we do not need to recompute the Fourier transforms at each scale, since using the scaling law [12] we arrive at the Fourier transform of a larger scale from that of a smaller one, thus further improving run-time speed.

## 6 Merging and Arbitration of Multiple Detections

As with most other face detection algorithms, it is necessary to post-process the raw classifier output in order to obtain clean results. We use a procedure similar to that in [4] and [13]. Correct detections of faces in images, as located by the algorithm in Section 5, tend to occur in multiple nearby pixel locations and multiple adjacent scales. On the other hand, false detections tend to be isolated in space, scale, or both. If we think of the raw detector output as a point process in the 3-dimensional location-scale space, then it makes sense to discard as false detections any faces for which the local intensity of the process is below a given threshold. While we could use a kernel density estimator to approximate the process intensity, this approach is unnecessarily complex, since we only need to know the intensity at the locations/scales classified as faces. We obtain the same result (as with a box kernel in location-scale space) with faster speeds as follows. For each (raw) detection  $F_i, i = 1, \dots, n$ , with  $x$ - $y$ -scale coordinates  $(x_i, y_i, s_i)$  we compute the coordinate-wise distances to every other detection

$$x_{i,j} = x_i - x_j, \quad y_{i,j} = y_i - y_j, \quad s_{i,j} = s_i - s_j. \quad (13)$$

The intensity at detection  $F_i$  is now estimated as  $c_i = |C_i|$ , where

$$C_i = \{F_j : |x_{i,j}| \leq \alpha(s_i, s_j) \text{ and } |y_{i,j}| \leq \beta(s_i, s_j) \text{ and } |s_{i,j}| \leq \gamma\}, \quad (14)$$

and  $\alpha, \beta$  and  $\gamma$  are predefined thresholds. We set  $\alpha(s_i, s_j) = \beta(s_i, s_j)$  to equal ten percent of the average detection window size at scales  $s_i$  and  $s_j$ , and  $\gamma = 2$ . Now, all detections whose estimated intensity  $c_i$  is below a given threshold (four in our case) are discarded as false alarms.

Remaining detections are further processed to eliminate overlaps. We process the list of detections in descending or-



Figure 5: Left: raw classifier output. Right: detections after thresholding, merging and arbitration of overlapping detections.

der of their estimated intensities. For a given detection  $F_i$ , we remove from the detection list all other elements of  $C_i$ , which necessarily have lower intensity since we are processing the list in order. Then we proceed to the next remaining detection and eliminate its lower ranked overlaps, and so on until all detections have been either processed or deleted. Finally the remaining detections are sorted once again by intensities and any remaining overlaps (beyond the  $\alpha, \beta, \gamma$  tolerances) are removed in the same fashion. Figure 5 shows an example of this procedure, with the left image showing the raw detections and the right image displaying the final result.

## 7 Experimental Results

We implemented the detector described in Section 5 on a Pentium III 1Ghz computer with 1Gb of RAM, and trained it on the data described in Section 2. Our error tolerance on the face class was set at 0.2% for all sub-stages. While the exact running speed of the system depends on the image and the specific training, we observe speeds of up to 12 frames per second on average, for uniformly sized  $320 \times 240$  pixel images. Some example results from images in the CMU/MIT test set are shown in Figure 6, where we can see, in addition to correct detections, some false alarms and missed faces.

## 8 Conclusions

We introduced a new face detection algorithm based on the CCCD classifier. In order to meet the performance requirements of the face detection task, we adapted the standard CCCD classifier, and developed it into a boosted tree structure. By boosting the classifier we overcome the difficulties associated with the severely skewed priors that are common in object detection problems. The resulting CCCD classifier

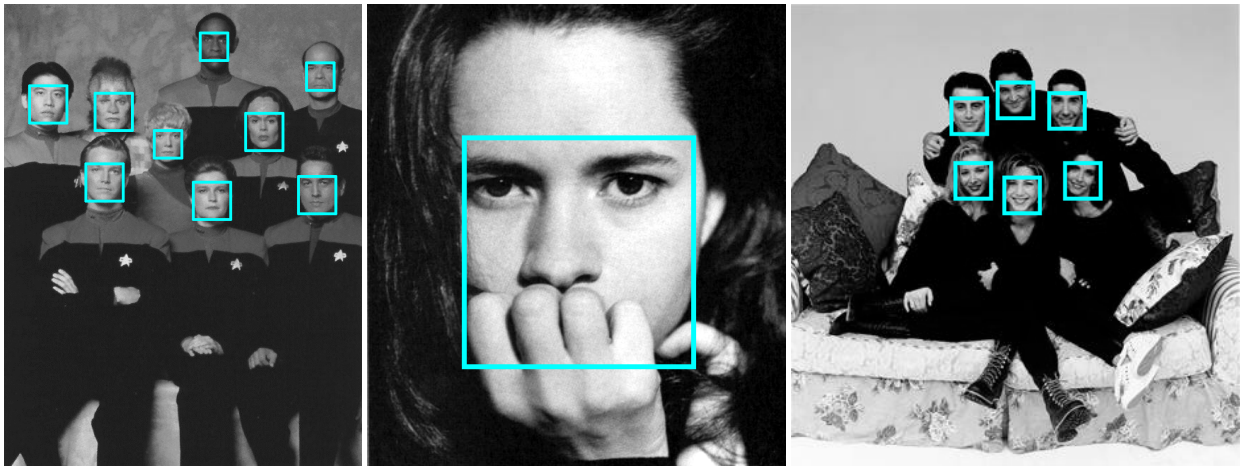
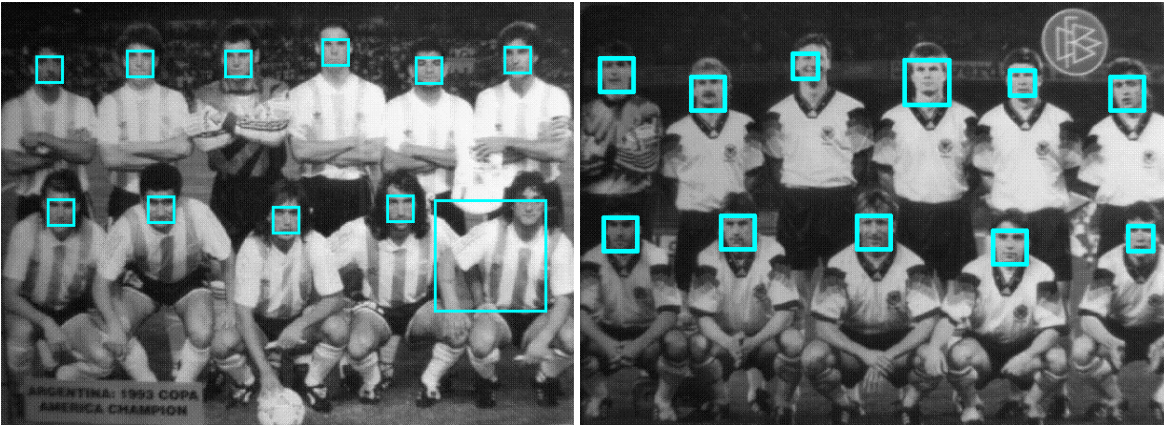
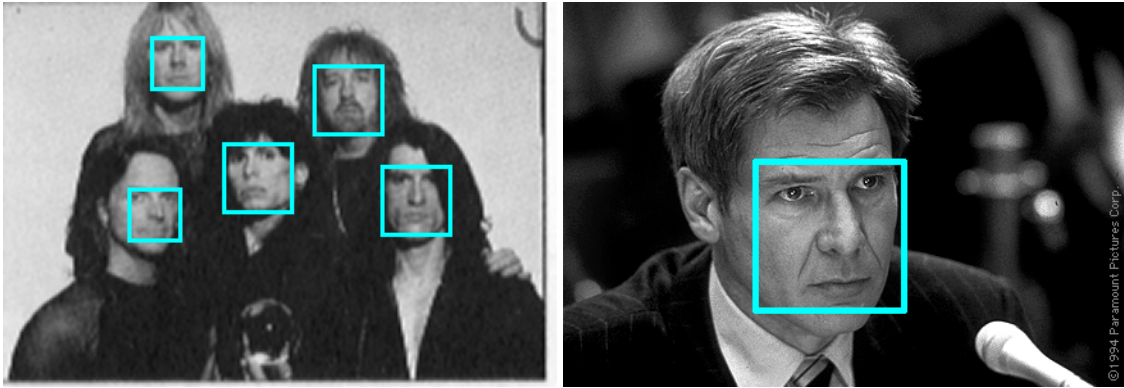


Figure 6: Example results from the CMU/MIT test set.

is of independent interest, and it will be further pursued for its application to general pattern recognition problems.

Thanks to the application of fast Fourier transforms in the computation of Euclidean distances, we are able to implement the boosted tree classifier for near real-time operation. The runtime speeds achieved are somewhat lower than those reported by the fastest algorithm in the literature [17, 18].

There are some outstanding issues with the current method, as well as some areas where we see a clear path for improvement. The choice of a the number of face prototypes for a given stage of the classifier is currently done heuristically. It would be very satisfying to have an automated method for choosing the appropriate complexity of the face-class representation. This might be done by simply truncating the greedy covering algorithm of the standard CCCD classifier based on some criterion related to runtime speed. It seems more efficient altogether to increase the complexity of the face-class representation by partitioning the data space and allowing the tree to have multiple branches at each level, corresponding to different cells. We are currently pursuing such an approach, the results of which will be reported elsewhere. Another avenue of active investigation is the use of different dissimilarity measures at different stages of the tree. While we are constrained by the use of Fourier transforms to use the Euclidean metric at the first sub-stage of the classifier, we have complete freedom after that. We can take advantage of this freedom in the space partitioning case under current development, by locally varying the dissimilarity function. We expect such a technique to yield higher performance with no runtime penalty. In addition, we plan to explore features other than the normalized image grayvalues for classification, with the expectation that they would yield a more invariant representation of the face class, especially in the presence of unconstrained illumination. In particular, we will consider the feature set used in [17], since we already compute the tables in Equation (10) as part of our algorithm.

## References

- [1] J. DeVinney C. Priebe, D. Marchette and D. Socolinsky. Classification using class cover catch digraphs. Technical Report 628, Department of Mathematical Sciences, The Johns Hopkins University, Baltimore, MD 21218, 2002. Submitted for publication.
- [2] F. Fleuret and D. Geman. Coarse-to-fine face detection. *IJCV*, 41(1/2):85–107, January 2001.
- [3] K. Fukunaga. *Introduction to Statistical Pattern Recognition*. Academic Press, San Diego, 2nd edition, 1990.
- [4] S. Baluja H. A. Rowley and T. Kanade. Neural network-based face detection. *PAMI*, 20(1):23–38, January 1998.
- [5] Q. Chen H. Wu and M. Yachida. Face detection from color images using a fuzzy pattern matching method. *PAMI*, 21(6):557–563, June 1999.
- [6] J. Mason J. Brand and M. Pawlewski. Face detection in colour images. In *ICIP01*, page Face Detection and Recognition, 2001.
- [7] D. Marchette J. DeVinney, C. Priebe and D. Socolinsky. Random walks and catch digraphs in classification. Technical report, Department of Mathematical Sciences, The Johns Hopkins University, Baltimore, MD 21218, 2002. To appear in *Proceedings of Interface*.
- [8] M. David J.C. Terrillon and S. Akamatsu. Automatic detection of human faces in natural scene images by use of a skin color model and of invariant moments. In *AFGR98*, pages 112–117, 1998.
- [9] J. P. Lewis. Fast template matching. In *Vision Interface*, pages 120–123, 1995.
- [10] D. J. Kriegman M. H. Yang and N. Ahuja. Detecting faces in images: A survey. *PAMI*, 24(1):34–58, January 2002.
- [11] M. Abdel-Mottaleb R.L. Hsu and A.K. Jain. Face detection in color images. *PAMI*, 24(5):696–706, May 2002.
- [12] W. Rudin. *Real and Complex Analysis*. McGraw Hill, New York, third edition, 1987.
- [13] B. Fasel S. Ben-Yacoub and J. Luttin. Fast face detection using MLP and FFT. In *AVBPA99*, 1999.
- [14] G. Lugosi S. Kulkarni and S. Venkatesh. Learning pattern classification—a survey. *IEEE Transactions on Information Theory*, 44(6):2178–2206, October 1998.
- [15] K. Sobottka and I. Pitas. Face localization and facial feature extraction based on shape and color information. In *ICIP96*, page 19A8, 1996.
- [16] K. K. Sung and T. Poggio. Example-based learning for view-based human face detection. *PAMI*, 20(1):39–51, January 1998.
- [17] P. Viola and M. Jones. Rapid object detection using a boosted cascade of simple features. In *CVPR01*, pages 1:511–518, 2001.
- [18] P. Viola and M. Jones. Robust real-time face detection. In *ICCV01*, page II: 747, 2001.