5-2012

Composite Feature-Based Face Detection Using Skin Color Modeling and SVM Classification

Swathi Rajashekar
Utah State University

Composite Feature-Based Face Detection Using Skin Color Modeling and SVM Classification

by

Swathi Rajashekar

A report proposal submitted in partial fulfillment

of the requirements of the degree

of

MASTER OF SCIENCE

in

Computer Science

Approved:

Xiaojun Qi Major Professor	Curtis Dyreson Committee Member

Stephen J. Allan
Committee Member

UTAH STATE UNIVERSITY

Logan, Utah

2012

ABSTRACT

Composite Feature-Based Face Detection Using Skin Color Modeling

and SVM Classification

by

Swathi Rajashekar, Master of Science

Utah State University, 2012

Major Professor: Dr. Xiaojun Qi
Department: Computer Science

This report proposes a face detection algorithm based on skin color modeling and support vector machine (SVM) classification. Said classification is based on various face features used to detect specific faces in an input color image. A YCbCr color space is used to filter the skin color pixels from the input color image. Template matching is used on the result with various window sizes of the template created from an ORL face database. The candidates obtained above, are then classified by SVM classifiers using the histogram of oriented gradients, eigen features, edge ratio, and edge statistics features.

ACKNOWLEDGMENTS

I would like to thank my major professor, Dr. Xiaojun Qi, for giving me the opportunity to be a part of this project, and for her invaluable support, guidance, and assistance during my work. I give my appreciation to my committee members, Dr. Curtis Dyreson and Dr. Stephen Allan, for their help.

I would like to thank my family and my friends for their continuous support and encouragement, without which this project would not have been possible. Lastly, I would like to thank my friend Dhaval Deshpande for his invaluable support and interest in my project.

Swathi Rajashekar

CONTENTS

Page

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1

INTRODUCTION

In recent years, research on face recognition has rapidly increased in the fields of computing and neuroscience. The potential for applications in computer vision communication and automatic access control is enormous [2]. Face detection is an important part and the first step for face recognition. The purpose of face detection is to localize and extract the face region from the background. However, there are lots of challenges involved in face detection, as the human face has a high degree of variability in appearance, illumination, distance from the image device, occlusion, rotation of head in different places, facial expression, and many more.

Face detection techniques can be roughly classified into four categories [8, 10], namely, skin color model-based approaches, template matching-based approaches, feature-based approaches, and statistical model-based approaches. Usually, face detection techniques integrate some or all of the four approaches to achieve high face detection accuracy and a low false detection rate. Here, we briefly explain each of the four approaches.

1.1 Skin Color Model-Based Approaches

Skin color model-based approaches build a skin color model using Gaussian normal distribution since color is one of the most widely used visual features in face detection. Specifically, said models convert the color image into an appropriate color space, such as HSV, YCbCr, or YIQ, to find skin color. These color spaces are more robust to the lighting conditions than the RGB color space and therefore are suitable for

face detection under different illuminations. The mean and covariance matrix of the skin color are then computed from the skin colors. Finally, the results of this computation are used to find the likelihood that each pixel in the input image is, indeed, a skin color.

1.2 Template Matching-Based Approaches

Template matching based approaches are commonly used to obtain regions with the highest potential to be faces. These methods perform pixel intensity comparison between a predefined template image and sub-regions of the image. A face template expresses a general appearance of a face and is related to face shapes [3]. Generally, averaging grayscale faces from a set of training images creates the grayscale face template. This template is matched with the potential candidates in sub-regions using normalized correlation. A predefined threshold is finally used to decide if the candidate is a face or a non-face.

1.3 Feature-Based Approaches

Feature-based approaches build upon explicit knowledge wherein features representing a face as defined by the designer are first extracted from images [4]. Face detection is thus achieved by verifying that with a certain degree of confidence. the features extracted from an image represent a face. Representative features are eigen features, Haar-like features, and edge features. In the following, we briefly explain each feature.

Eigenfaces have long been used for face detection and recognition purposes. Principal component analysis (PCA) is performed on a set of training face images to obtain the eigen vectors, which are called as eigenfaces. The projections of the mean

adjusted face images along the eigen vectors are used as the eigen features for training and classification purposes.

Haar-like features are the digital image features used in object recognition. Viola and Jones [9] were the first to use Haar wavelets as a basis for developing Haar-like features. A Haar-like feature considers adjacent rectangular regions at a specific location in a detection window, sums up the pixel intensities in these regions, and calculates the difference between them. This difference is then used to categorize subsections of an image. Given that in all faces the region of the eyes is darker than the region of the cheeks, a common Haar-like feature for face detection is a set of two adjacent rectangles that lie above the eye and the cheek region. The position of these rectangles is defined relative to a detection window that acts like a bounding box to the face. A window of the target size is moved over the input image, and the Haar-like feature is calculated for each subsection of the image. This difference is then compared to a learned threshold to separate non-faces from faces. The key advantage of a Haar-like feature over most other features is its calculation speed, since a Haar-like feature of any size can be computed in a constant time by using integral images. The integral images can be defined as two-dimensional look up tables in the form of a matrix with the same size of the original image. Each element of the integral image contains the sum of all pixels located on the upper-left region of the original image. This integral image uses only four lookups to compute the sum of rectangular areas in the image, at any position or scale.

Edge features are used to compute the edges of the image. The canny edge detector, Roberts edge detector, and Sobel edge detector are commonly used to extract

edge features. These edges normally represent the contour lines of the face, including eyes, eyebrow, lips, nose, and face boundary.

1.4 Statistical Model Based Approaches

Statistical model-based approaches use a statistics-based classifier, such as a support vector machine (SVM), to classify the candidate region as a face or a non-face. Specifically, the SVM trains a classifier by solving the optimization problem to decide which instances of the training data set are support vectors. These support vectors are the necessarily important instances to form the SVM classifier [6]. The learned SVM classifier is further employed in any classification task.

CHAPTER 2

RELATED WORK

Much work has been done, and many approaches have been proposed to perform face detection. Here, we briefly review several such works that are closely related to the proposed work.

Alajel et al. propose a face detection algorithm [1] using skin color modeling and the modified Hausdorff distance. They first calculate the probability of a pixel as being a skin color based on the mean and the covariance matrix of learned skin colors. A predetermined threshold is used to further decide skin color pixels. The authors then perform a series of morphological operations to remove the noise and separate potential face candidates. Finally, they apply a template-based object classifier to classify the skin color candidates as a face or a non-face using the modified Hausdorff distance, which is computed on the Sobel edges of the template and the potential face candidates. A predetermined threshold is also applied to decide whether the skin color candidate is a face or a non-face. Their experimental results on 160 images with 109 faces show a correct detection rate of 87.5% and a false detection rate (error rate) of 4.59%.

Li et al. [7] propose a novel face detection method by combining the skin color detection and an improved AdaBoost algorithm. They first apply the commonly used skin color model and the morphological operations to detect the significant skin color features in the images. This step functions as a filter to narrow the search space and speed up the detection of faces from complex background. They then extract simple rectangular features, called Haar features [9], to represent each candidate patch. They finally apply an improved AdaBoost algorithm to find the face candidates within each patch. The

original AdaBoost algorithm [9] constructs a strong classifier as a linear combination of weak classifiers. It is a self-adaptation iterative algorithm. Specifically, it selects the most important features from a big feature candidate set and makes a weak classifier for the selected features. It then combines the multiple weak learners to construct a strong learner.

Li et al. improve the original AdaBoost algorithm by first categorizing the training sample set as face and non face images with values of 1's and 0's, respectively. A normalized weight is assigned to each training sample based on the category. For each feature, this improved AdaBoost algorithm trains its weak classifier to determine its threshold and offset. After all the weak classifiers are determined, the weak classifier with the minimum error rate is considered to update weights of the training samples and build a strong classifier. A cascade of such strong classifiers is finally used to quickly eliminate the background area. Their experimental results on 66 images with 350 faces show a correct detection rate of 92.86% and a false detection rate of 1.5%.

Kherchaoui and Houacine [10] use the Gaussian mixture-based skin color model together with geometrical face characteristics to detect faces. They consider detected skin regions as face candidates based on a set of geometrical constraints. For example, they use the number of holes in the connected region as one of the verification steps, as a face admits non-skin color pixels only at eye and mouth regions. In addition, they use the region orientation, region size, and the region dimension ratio [10] to filter out the non-face. To this end, they consider the candidate as a potential face if the ratio of width and height of a region is in the range 0.8 to 1.8. Finally, they apply template matching to reach the final decision depending on the cross-correlation-based similarity between the

candidate face region and the face template, which is created by averaging multiple grayscale faces. Their experimental results on 182 faces of GTAV database show a correct detection rate of 79.11% including correct partial detection and a false detection rate of 21.42%.

Shavers et al. [11] present a face detection algorithm based on SVMs. Support vectors, the archetypes for face and non-face images, are generated, and the λ coefficients for these support vectors are determined from a set of training images using the multiplicative update algorithm [12]. The SVM algorithm, which is now programmed and trained according to the support vector archetypes, maps the test images into a higher dimension transform space wherein a hyperplane decision function is constructed. The hyperplane decision function is based on the polynomial kernel function with degree one. The decision function is constructed equidistance between support vector archetypes to give an optimal hyperplane decision function. The aforementioned SVM is applied to a 20×20 window of pixels (400 dimension vector) extracted from the 80% of the Olivettie Research Lab (ORL) face training images and tree, door, window non-face training images and to learn the optimal hyperplane decision function for the face classification. Their experimental results on the remaining 20% ORL face images show a correct detection rate of 95%. Further, they do not report any false detection rate.

Paul and Gavrilova [13] present a principal component analysis- (PCA) based modeling of the geometric structure of the face for automatic face detection. They first perform Gaussian mixture-based skin color modeling to remove the background of the image from the skin regions. They next apply a region-growing algorithm to generate black and white templates for skin regions. Finally, they improve the performance of the

template matching by applying canny edge detection on the geometric structure model of the face candidate produced by PCA. Specifically, they apply PCA to obtain the first K-dominant principal components to project the skin regions. They then reconstruct the skin regions using the first K-dominant principal component. These reconstructed skin regions effectively reduce noise and balance the intensity of skin regions. Finally, they apply the canny edge detector on the reconstructed skin regions to obtain the face boundary. The obtained face boundary is further compared with the face boundary of the template after applying an oval shaped mask to decide whether the region is a face or a non-face based on a predetermined threshold. Their experimental results on CIT, BaoFace, Essex, and Georgia Tech databases show a correct detection rate of 98.7%, 97.1%, 97.1%, and 85.2%, respectively. Moreover, they do not report the false detection rate.

All these face detection algorithms are capable of detecting faces in their chosen images at a decent correct detection rate. However, we observe that face detection algorithms [1, 7, 10, 12, 14] do not address the issues when the candidate patch contains multiple faces. For example, Shavers et al. [11] exclusively use input images containing only one face for their experiments. To address this problem and to improve overall face detection accuracy and efficiency, we propose a novel face detection technique integrating skin color modeling, template matching, feature extraction, and SVM classification.

Firstly, the image is segmented based on a skin color model in the YCbCr color space. Secondly, a series of morphological operations are further employed to remove small connected components that are unlikely to be a face, keep each candidate face

isolated, and connect the holes created inside a face for eyes and lips which are not of skin color. Each resultant connected component is used as a mask to extract the potential candidate from the original image. Thirdly, a novel template matching technique is applied on each resultant connected component to locate possible multiple faces within the connected component. Fourth, a set of composite features, namely, eigen features, histogram of oriented gradients (HOG), and edge-based statistics, are extracted from the candidate faces remaining from the previous step. Fifthly and finally, these features are further fed into the SVM to finally classify the candidate faces as true faces.

CHAPTER 3

THE PROPOSED FACE DETECTION ALGORITHM

The proposed face detection algorithm has four main components.

- Skin color segmentation
- Template matching
- Feature extraction
- SVM classification

Figure 1 shows a block diagram of the proposed online face detection system. In the following sections, I explain each component in detail.

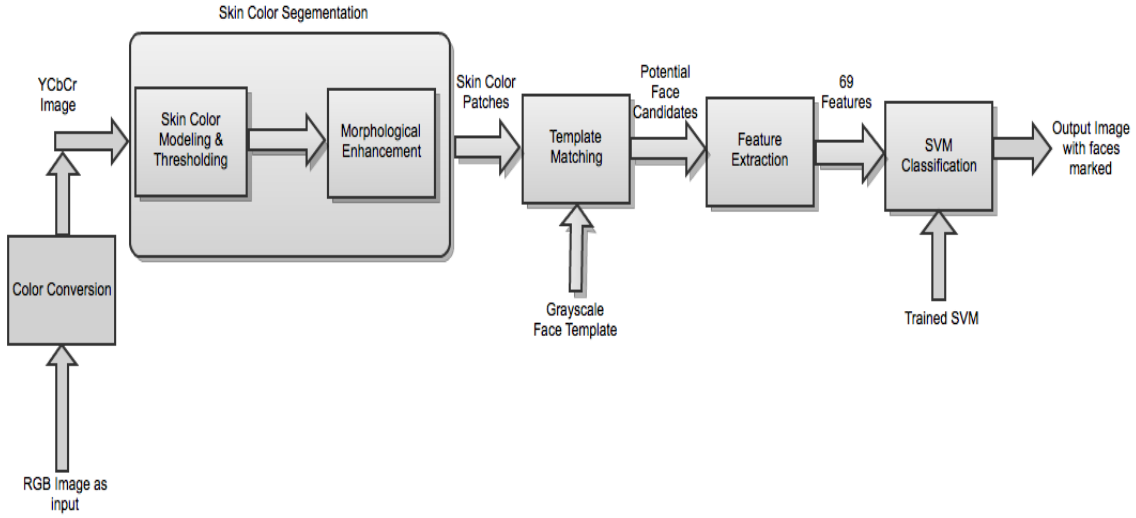

Figure 1: Block diagram of the proposed face detection system.

3.1 Skin Color Segmentation

The skin color segmentation process filters out all the non-skin color from the input image, keeping only skin color, since a primary method of detecting faces is through their color. This process has two prominent components, namely,

- Skin color modeling and thresholding
- Morphological enhancement

3.1.1 Skin Color Modeling and Thresholding

The input color image is typical in the RGB format. However, RGB components are subject to lighting conditions, and face detection may fail if, lighting conditions change. As a result, we use Equation (1) to convert RGB components into YCbCr components for removing the effect of luminosity during our image processing process.

$$\mathbf{Y} = 0.299\mathbf{R} + 0.587\mathbf{G} + 0.114\mathbf{B}$$

$$\mathbf{Cb} = -0.169\mathbf{R} - 0.332\mathbf{G} + 0.500\mathbf{B} \quad \textbf{(1)}$$

$$\mathbf{Cr} = 0.500\mathbf{R} - 0.419\mathbf{G} - 0.081\mathbf{B}$$

In the YCbCr components, the luminance (brightness) information is contained in a Y component, and the chrominance information is contained in the Cb (blue) and Cr (red). The Cb and Cr components are independent of luminosity and give a good indication on whether a pixel is part of skin or not. In addition, the Cb and Cr values for various skin colors fall in the blue regions shown in Figure 2 [16].

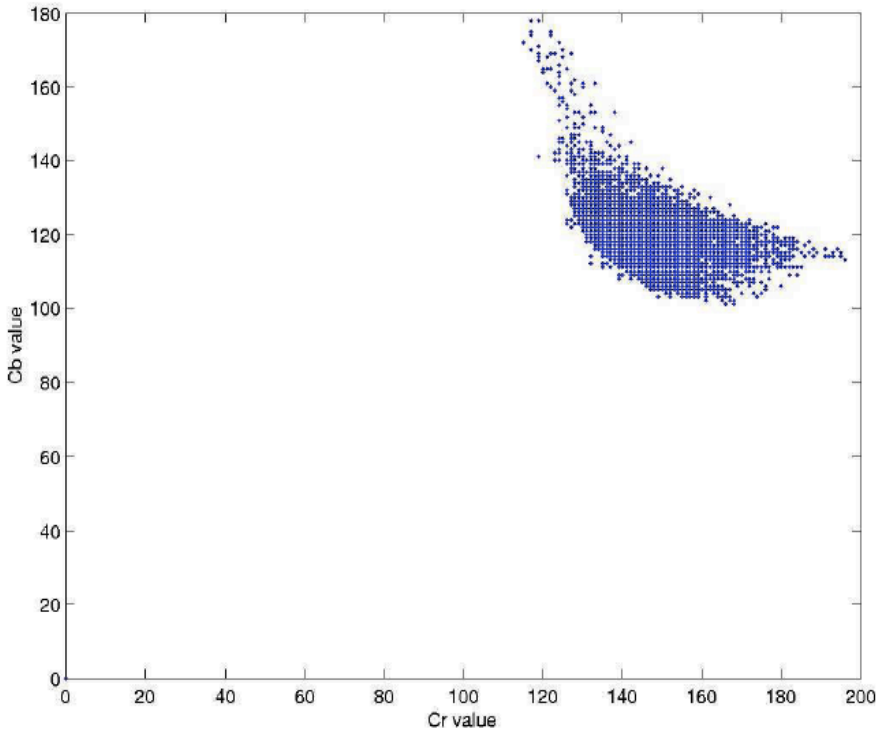

Figure 2: Strong correlation between Cr and Cb values for skin pixels [16].

It is evident that background and faces can be distinguished by applying maximum and minimum threshold values for both Cb and Cr components. In our system, we consider a pixel to be skin, if its Cb and Cr values fall in the range shown below.

$$100 <= Cb <= 140$$

$$140 <= Cr <= 165$$

In other words, pixels whose intensities fall in the above range are marked as 1's, and the other pixels are marked as 0's. A binary image with all skin color represented by a 1 is created. However, other parts of the body, such as exposed arms, legs, and other skin color background objects can also be captured. A post-processing step (e.g.,

morphological enhancement) must be employed to extract all the skin objects separately for future classification.

Figure 3 shows the output binary image after skin color modeling and thresholding. We can clearly see that all the skin colors including the face, arm, and part of the brownish hair are marked with 1's and non-skin colors are marked as 0's. These nonface parts are eliminated in a later process.

(a) (b)

Figure 3: Illustration of skin color modeling and thresholding: (a) Original color image; (b) Skin color modeling and thresholding result.

3.1.2 Morphological Enhancement

We perform a series of morphological operations as a post-processing step to extract all the skin objects separately for future classification. Figure 4 summarizes all the components in the process. In the following, I explain each component in detail.

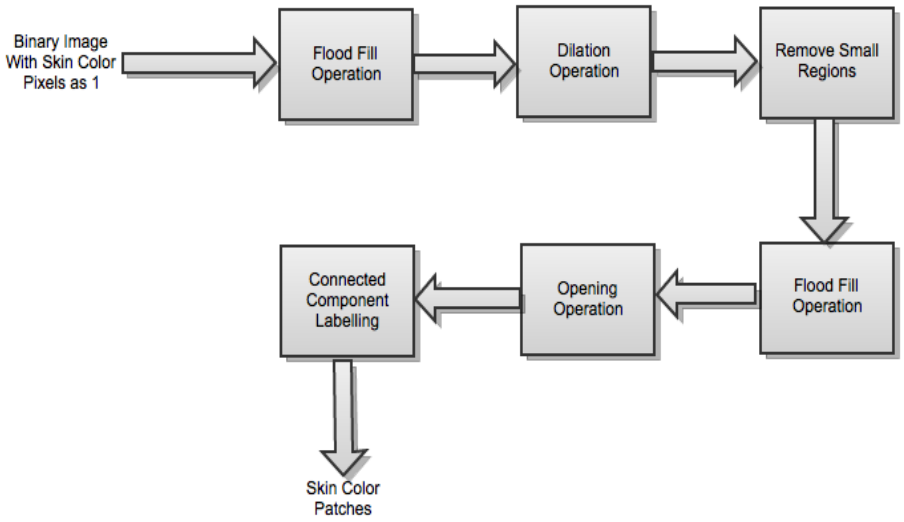

Figure 4. Block diagram of morphological enhancement.

Flood Fill Operation. We perform a flood-fill operation on the binary image to fill the holes created inside the face region, because of the eyes and lips as shown in Figure 3 or the holes created at the boundary after applying the dilation operation. These holes tend to separate all the objects from each other and therefore should be removed.

Dilation Operation. We apply dilation operation on the binary images with holes, using a disk structuring element of radius of 1. This operation fine-tunes the boundary of the regions by connecting small breaks and enlarges the shapes a bit for further processing.

Remove Small Regions. We remove the areas whose area is less than 200 pixels since they are too small to be a face region.

Opening Operation. We apply an opening operation using a disk structuring element of size 3 to remove connections between closely connected regions if any.

Connected Component Labeling. We apply the connected component labeling algorithm to obtain a separate component for each object. Ideally, we expect that each connected component contains at most one face. However, it is possible that the connected component may contain two or more faces in them. This issue is addressed in the template matching process later on.

Figure 5 shows the intermediate results for the morphological enhancement on the image shown in Figure 3(b). The results of the last three steps are omitted since they are the same as the results shown in Figure 5(c).

(a) (b) (c)

Figure 5: Intermediate morphological enhancement Results. (a) Holes removing results; (b) Dilation results; (c) Small components removal results.

3.2 Template Matching

Template matching performs a pixel intensity comparison between a predefined template image and sub-regions of the image. Template matching can be performed using convolution or correlation. In our algorithm, we use correlation to match the template

image and the candidate of interest.

We use the ORL face database [17], which contains all grayscale faces, to obtain a grayscale face template by averaging all the face images in the database. In this ORL database, we have faces with different expressions, different illuminations, and different orientations. There are ten different images for each of 40 distinct subjects. The size of each image is 192×168 pixels with 256 gray levels. The grayscale face template image created from the ORL face database is shown in Figure 6.

Figure 6: Grayscale face template created from ORL face database.

Correlation is a measure of the degree to which two variables agree, not necessarily in actual values but in general behavior. The two variables are the corresponding pixel values in two images: grayscale face template and the sub-region of the input image. Correlation between the template and the sub-region of the input image (i.e., skin color patch) is calculated using Equation (2).

$$cor = \frac{\sum_{i=0}^{N-1}\left(x_i - \bar{x}\right)\cdot\left(y_i - \bar{y}\right)}{\sqrt{\sum_{i=0}^{N-1}\left(x_i - \bar{x}\right)^2 \cdot \sum_{i=0}^{N-1}\left(y_i - \bar{y}\right)^2}} \qquad (2)$$

where,

- x is the template gray level image
- $\bar{x}$ is the average grey level in the template image
- y is the source image section
- $\bar{y}$ is the average grey level in the source image
- N is the number of pixels in the section image with N equaling the template image size
- cor is the correlation value between –1 and +1, with larger values representing a stronger relationship between the two images.

Since a patch (e.g., the minimum bounding box to cover the connected component) may contain multiple faces and the size of the faces may vary significantly in the input images, we perform the correlation operation on the patch iteratively with six different scaled grayscale face templates. Specifically, the template sizes are varied from 90×79, 80×70, 70×61, 60×52, 50×43, to 40×35, maintaining the same ratio of rows to columns (192:168) as the grayscale face template. Figure 7 summarizes the algorithmic view of this operation.

```
Scale the grayscale face template of size 112×92 to the size of 90×79
While the height of the patch > 90 pixels and the height of scaled template > 30 pixels
   Position the left corner of the grayscale face template at each pixel within the patch
   Compute the correlation value at each pixel using Equation (2)
   Find the maximum correlation value MaxCor within the patch
   Record the location (Xmax, Ymax) with MaxCor
   If MaxCor > threshold (e.g., 0.32)
      The box of the same size as the investigated grayscale face template, whose left corner
      starting at (Xmax, Ymax) is considered containing a potential face
      Set all values in the box as -1's to make this box unlikely chosen in the next iteration
   Else
      Reduce the template height by 10 pixels while keeping the same aspect ratio
   Endif
 Endwhile
```

Figure 7: Algorithmic view of the iterative template matching process.

(a) (b) (c)

Figure 8: Illustration of iterative template matching results. (a) Original patch obtained from the skin color modeling and thresholding; (b) The candidate box obtained by matching with the 1st scaled template of 90×79; (c) The candidate box obtained by matching with the 2nd scaled template of 80×70.

Figure 9: Two face candidates obtained from iterative template matching.

Figure 8 demonstrates the template matching result for one single patch with multiple faces. Figure 9 shows the face candidates corresponding to the two candidate boxes.

3.3 Feature Extraction

Four kinds of complementary features are extracted from the candidates obtained from iterative template matching. We consider four main features of face, namely,

- Histogram of oriented gradients (HOG) features
- Eigen features
- Edge ratio
- Edge statistics

3.3.1 HOG Features

HOG features [5], which describe an object's appearance and shape by the distribution of intensity gradients or edge directions, have recently been widely used for object, pedestrian, and face detection. These features count the occurrences of gradient orientation in localized portions of a given image. First, local histograms of image gradient orientations in a dense grid are computed. This grid divides the image into small uniformly spaced special regions called cells. Second, accumulate a local 1D histogram of gradient directions or edge orientations over all pixels in each cell to form the HOG representation. Third, the local responses are contrast normalized by normalizing all the cells using the accumulated local histogram over slightly larger regions called blocks. Referred as HOG descriptors, these normalized descriptor blocks are invariant to illumination or shadowing. The step-by-step process of computing HOG features follows:

1) Compute the gradient values by simply applying the 1-D centered, point-discrete derivative mask in one or both of the horizontal and vertical directions.

Specifically, this step requires filtering the color or intensity data of the image with the following filter kernels: [-1 0 1] and $[-1\ 0\ 1]^T$.

2) Create the cell histograms by casting the weighted vote of each pixel on an orientation-based histogram channel established on the values found in the gradient computation. The cells themselves can either be rectangular or radial in shape, and the histogram channels are evenly spread over 0 to 180 degrees or 0 to 360 degrees, depending on whether the gradient is unsigned or signed.
3) Locally normalize the gradient strengths to account for changes in illumination and contrast by grouping the cells together into larger, spatially connected blocks. The HOG descriptor is then the vector of the components of the normalized cell histograms from all of the block regions. These blocks typically overlap, meaning that each cell contributes more than once to the final descriptor.

In our implementation, we consider 3×3 cell blocks and 4 histogram bins for each block, thus having a 36×1 HOG feature for each input candidate patch. That is, the histogram for each cell block is normalized before putting it to the HOG feature vector. If 'H' is a 4×1 non-normalized histogram vector obtained for a given cell block, the normalized histogram is computed by Equation (3),

$$\text{NormalizedH} = H(k) / (\text{norm}(H) + e) \tag{3}$$

where,

- k ranges from 1 to 4, which are the bin numbers
- norm(H) is a function that calculates vector norm for the histogram vector
- e is some very small constant.

Appending all the normalized histogram for each bin gives us the 36-dimensinoal (3×3×4) HOG features.

3.3.2 Eigen Features

We first scale the faces in the ORL database to the size of 40×35 since our minimum dimension of the candidate patch would be 40×35. We also scale all the face candidate patches to 40×35 before calculating its eigen features. The steps for computing the eigen features for an input candidate patch are summarized below.

1) Convert each scaled face image in the ORL dataset to a column vector of size 1400×1. Denote all face images as $D = \{x_1, x_2, x_3 \ldots x_M\}$, where x_1 represents the column vector of the first face image and M is the total number of images in the ORL dataset.
2) Compute the mean face μ of size 1400×1 and the covariance matrix C of size 1400×1400 by Equation (4).

$$\mu = \frac{1}{M}\sum_{m=1}^{M} x_m$$
$$C = \frac{1}{M}\sum_{m=1}^{M}[x_m - \mu][x_m - \mu]^T \quad (4)$$

3) Compute the covariance matrix C's eigen values (e.g., λ_k's) and corresponding eigenvectors (e.g., u_k's) using Equation (5).

$$Cu_k = \lambda_k u_k \quad (5)$$

4) Construct the matrix $U=[u_1\ u_2\ \ldots\ u_k]$ using k dominant eigen vectors. Specifically, u_1, u_2, and u_k, respectively, represent the eigenvectors of the k largest values λ_1,

λ_2.... λ_K, where $\lambda_1 > \lambda_2 > .... > \lambda_K$. These eigenvectors are mutually orthogonal and span a k-dimensional subspace called principal subspace.

5) Compute the eigen features of the scaled candidate patch by Equation (6)

$$\text{Eigen} = U^T (x - \mu) \quad (6)$$

where x represents the column vector of the scaled candidate patch, μ represents the mean face, U^T is the transpose of the matrix U, and Eigen represents the eigen features of the scaled candidate patch, which is a k×1 vector.

It should be noted that steps 1 through 4 are computed offline. The matrix U computed at step 4 and the mean face are stored in a file and loaded to compute the eigen features during the online face detection process. In our implementation, we choose 12 important eigenvectors to create the matrix U. Thus, the eigen feature is a 12×1 vector.

3.3.3 Edge Ratio

To compute this feature, we extract edges using the canny edge detector on the converted grayscale patch. We then divide the edge image into 4×4 non-overlapping blocks and compute the ratio of the edge pixels in each of the 16 blocks. This feature is a 16×1 feature vector and captures the edge distribution in each block. Edge ratio is an important feature since the edges for non-faces vary greatly compared to those of faces.

We choose the Canny edge detector to extract edges since it uses a multi-stage algorithm to detect a wide range of edges in the image. The Canny edge detector generally achieves good detection by marking as many real edges in the image as possible and reducing the false edges created by the image noise.

3.3.4 Edge Statistics

The edge statistics feature contains the statistics of the edge information. Edges are significant features in classifying faces and non-faces, so this component is important. To compute this edge statistics feature, we first apply the vertical Sobel edge detector as a filter to obtain a filtered patch that contains the first derivative changes along the vertical direction. We next compute the mean and standard deviation of the filtered patch. Similarly, we apply the horizontal Sobel edge detector as a filter to obtain a filtered patch that contains the first derivative changes along the horizontal direction. We then compute the mean and standard deviation of the filtered patch. Finally, we include the entropy of the Canny edge image. In total, we extract 5 values for the edge statistics feature vector. The vertical Sobel edge detector is shown in the following matrix:

$$\begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}$$

The horizontal Sobel edge detector is shown in the following matrix

$$\begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix}$$

3.4 Support Vector Machine (SVM) Classification

Support vector machines are a set of related supervised statistical learning methods that analyze data and recognize patterns. They are typically used for classification (machine learning) and regression analysis. Vapnik invented the original SVM algorithm, and the current standard incarnation (soft margin) was proposed by

Cortes and Vapnik [18].

The standard SVM is a non-probabilistic binary linear classifier. It uses a set of training samples marked as belonging to one of two categories to build a model that predicts whether a new sample falls into one category or the other. Intuitively, an SVM model finds a clear gap that is as wide as possible to separate training samples into two categories. New samples are then predicted to belong to a category based on which side of the gap they fall on. More formally, an SVM constructs a hyperplane or set of hyperplanes in a high-dimensional space for classification, regression, or other tasks. A good separation is achieved by the hyperplane that has the largest distance to the nearest training data points of any class (so-called functional margin), since in general the larger the margin the lower the generalization error of the classifier. Figure 10 shows the support vectors for the linearly separable samples. These support vectors are decided by three samples on the dashed line. The solid line represents the hyperplane for the decision function.

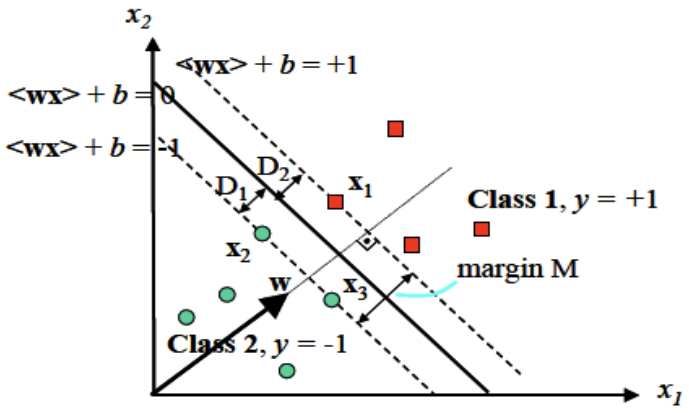

Figure 10: SVM for the linearly separable samples.

However, it often happens that the samples in the original dimensional space are not linearly separable. As a result, SVM schemes apply the kernel trick to map samples to a higher dimensional space such that the newly mapped samples are linearly separable. Figure 11 demonstrates this idea. The kernel trick ensures that cross products of two newly mapped samples may be computed easily in terms of the two samples in the original space. Thus, the computational load is reasonable.

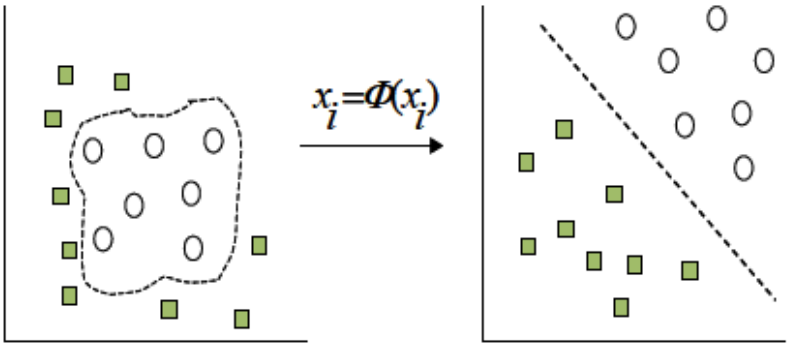

Figure 11: Non-linearly separable samples vs. linearly separable higher-dimensional samples.

Soft-Margin SVMs are commonly used to further allow mislabeled samples [18]. That is, if no hyperplane exists that can split the "yes" and "no" examples, the *Soft Margin* method will choose a hyperplane that splits the samples as cleanly as possible, while still maximizing the distance to the nearest cleanly split samples. Figure 12 illustrates the basic idea of the soft-margin SVMs.

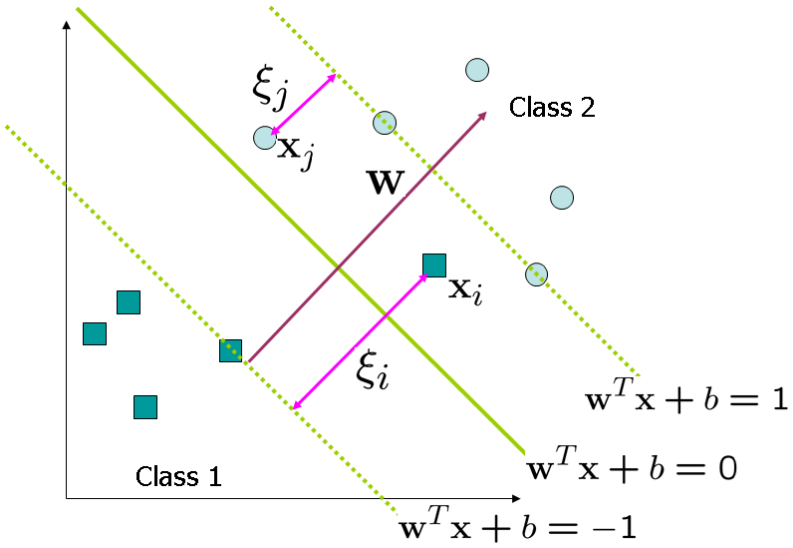

Figure 12: Illustration of soft-margin SVMs.

The soft-margin SVMs enforce the following two constraints to accommodate the mislabeled samples:

$$c_i\left(w\cdot x_i - b\right)\geq 1-\xi_i, \quad \xi_i \geq 0$$

Here, X_i is a p-dimensional feature vector (i.e., the input to the SVM), c_i is either 1 or -1 indicating the class (face or non-face) to which X_i belongs, ξ_I is a slack variable that measures the degree of misclassification of the datum X_i. The optimization becomes a trade-off between a large margin and a small error penalty. If the penalty function is linear, the optimization problem becomes:

$$\min_{w,\xi}\left\{\frac{1}{2}\|w\|^2 + C\sum_{i=1}^{n}\xi_i\right\}$$

where, C is the trade-off parameter between a large margin and a small error penalty. The two constraints along with the above objective can be solved using Lagrange multipliers. Its dual optimization problem becomes:

$$\min_{w,\xi,b}\max_{\alpha,\beta}\left\{\frac{1}{2}\|w\|^2 + C\sum_{i=1}^{n}\xi_i - \sum_{i=1}^{n}\alpha_i\left[c_i\left(w\cdot x_i - b\right) - 1 + \xi_i\right] - \sum_{i=1}^{n}\beta_i\xi_i\right\}$$

In this form, it is clear that the key advantage of a linear penalty function is that the slack variables vanish from the dual problem, with the constant C appearing only as an additional constraint on the Lagrange multipliers.

In our system, SVM is trained with 607 face patches and 424 non-face patches. These are obtained by applying iterative template matching on ORL face database images and many other images, which were not used for initial testing of the algorithm. For all these face and non-face candidate patches, we extract 69 features as described in Section 3.3 to form the input to the SVM. To enforce the kernel trick, we use a quadratic kernel function to map training samples in 69 dimensions to a higher dimension. We experiment on different values of C to decide on the best configuration for training the SVM. Figure 13 shows the block diagram of the offline training process.

After training SVM with the training samples, the trained SVM can be used directly to classify each candidate patch as a face or a non-face. To this end, we extract 69 features for each candidate patch and feed these features into the trained SVM for the classification task.

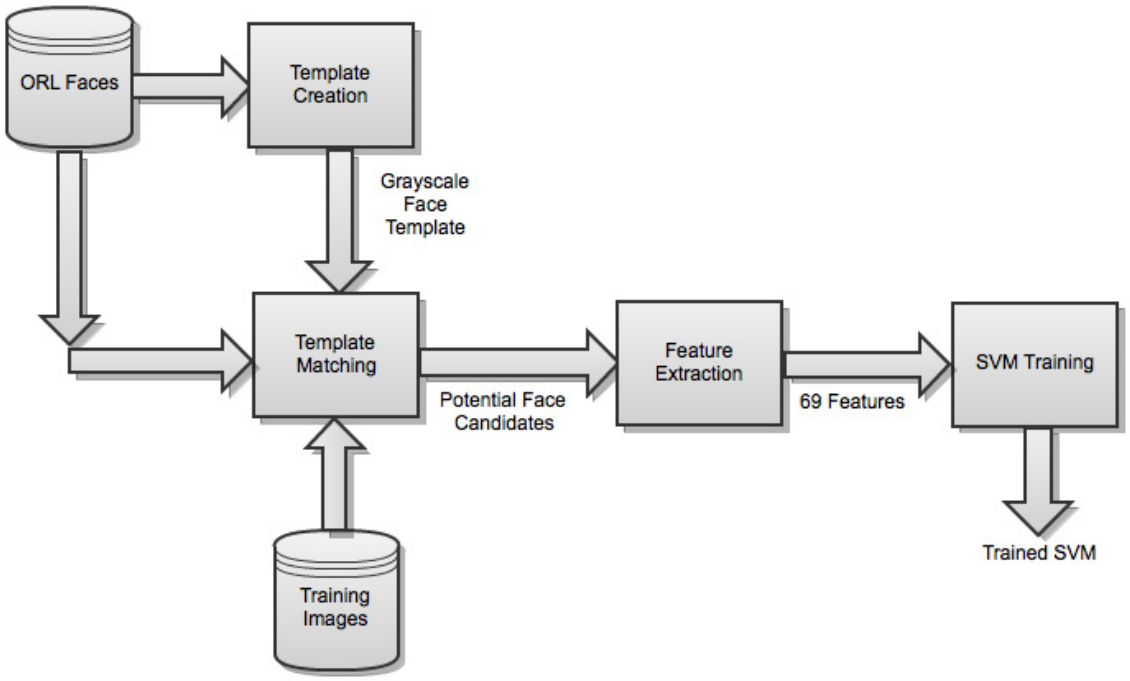

Figure 13: The block diagram of the offline Training process.

CHAPTER 4

EXPERIMENTAL RESULTS

Various experiments have been performed on a set of images to test the proposed face detection algorithm. Specifically, we choose 56 images with 258 faces for our experiments. These images belong to different categories, covering most scenarios in digital photographs. The following summarizes each category together with the number of test images in it. It should be noted that several images may belong to multiple categories.

- 32 images with multiple faces
- 24 images with single face
- 10 images with multiple faces that are close to each other
- 22 images with multiple faces that are far apart from each other
- 15 images with varied background
- 12 images with faces wearing spectacles
- 2 images with faces in less light

In our experiments, the total number of faces in all test images was 258. None of the faces were rejected from the skin color modeling and thresholding. The total number of non-faces falsely detected as faces from template matching was 369. The template matching also missed 15 faces out of 258 faces. Figure 14 and Figure 15 show the detected candidate faces after applying template matching and SVM classification on an image with multiple faces close to each other, respectively. It should be noted that there

Figure 14: Candidate patches detected after template matching

Figure 15: Final classified faces after SVM classification.

are a lot of connected skin color components after applying skin color modeling and thresholding. As a result, we do not show the results at this stage.

Figure 14 clearly shows that many of the non-face regions such as the hands and background were detected as faces during template matching. Three of the faces were missed during template matching. This is because those faces were partially covered by other faces that were detected. This partial covering makes detection of the covered faces difficult. Figure 15 clearly shows that all faces at the template matching stage are kept, and many of the non-faces detected as faces at the template matching stage are removed by SVM classification.

Figure 16 and Figure 17 show the template matching and SVM classification results for an image with an intricate background of a lot of skin color pixels, respectively. This test image also has a person wearing spectacles.

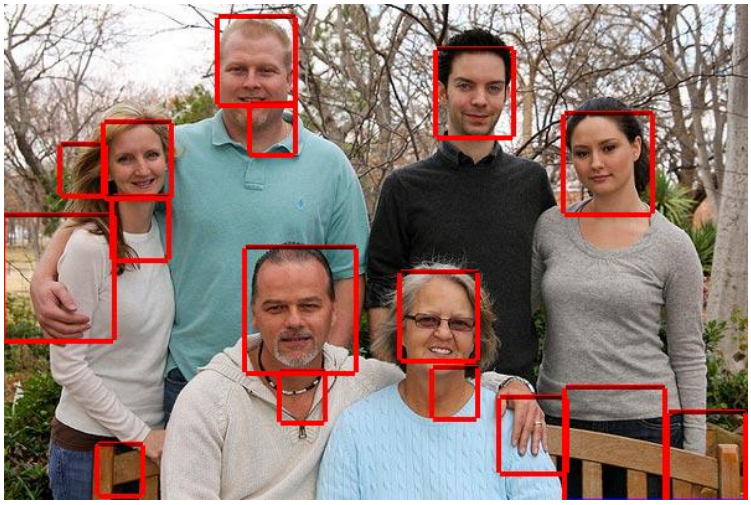

Figure 16: Candidate patches detected after template matching.

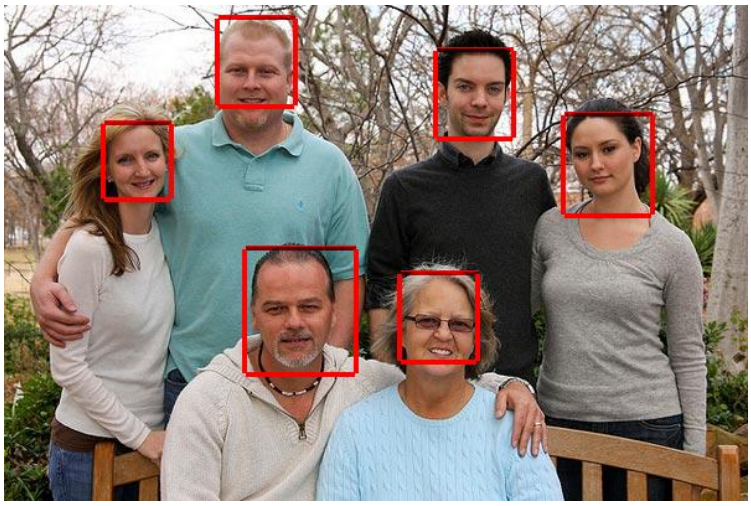

Figure 17: Final classified faces after SVM classification.

Figure 18 and Figure 19 show the template matching and SVM classifier output for an input image having multiple faces of different race, as well as having and an intricate background. We can see that a multiple number of non-faces are falsely detected as faces in template matching. But the SVM classifier does a great job in classifying them.

As mentioned in Chapter 3, our face detection algorithm uses four kinds of features, namely, HOG features, eigen features, edge ratios, and edge statistics, for SVM training and classification. We ran different experiments to summarize the performance under different features. The following summarizes those experiments.

First, we ran an experiment to decide whether normalized eigen features should be used in face detection since the other three features are in the range of [0, 1]. Table 1 compares face detection performance using eigen features combined with the three other features and normalized eigen features combined with the three other features. In Table 1, we also compare the face detection performance using 12 normalized eigen features and 25 normalized eigen features together with the other three features.

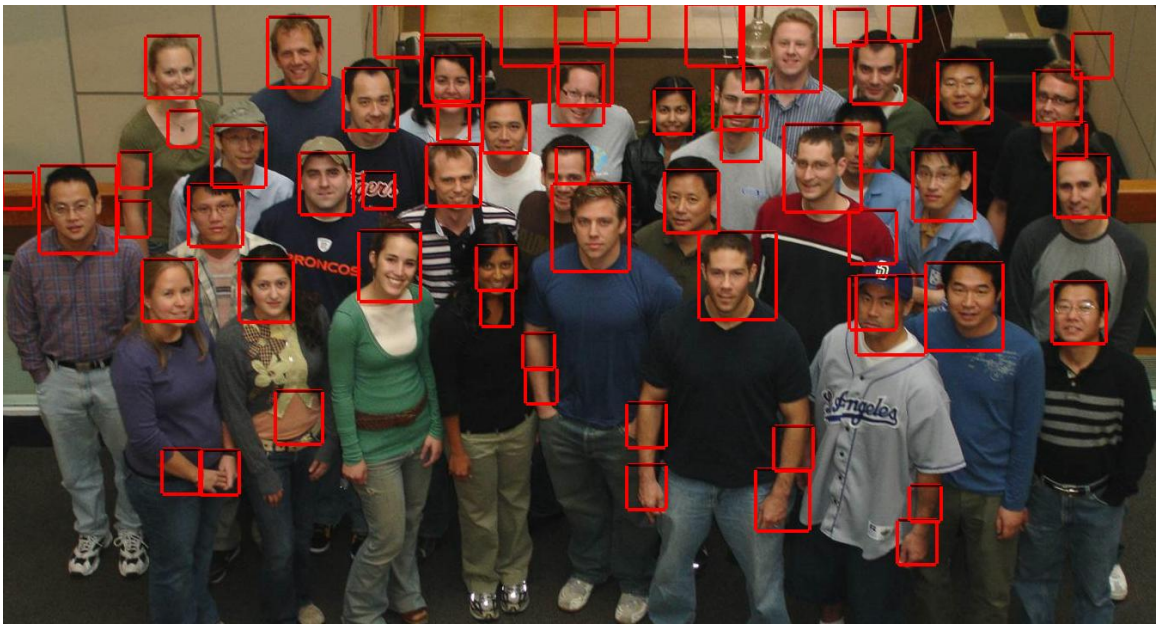

Figure 18: Candidate patches detected after template matching.

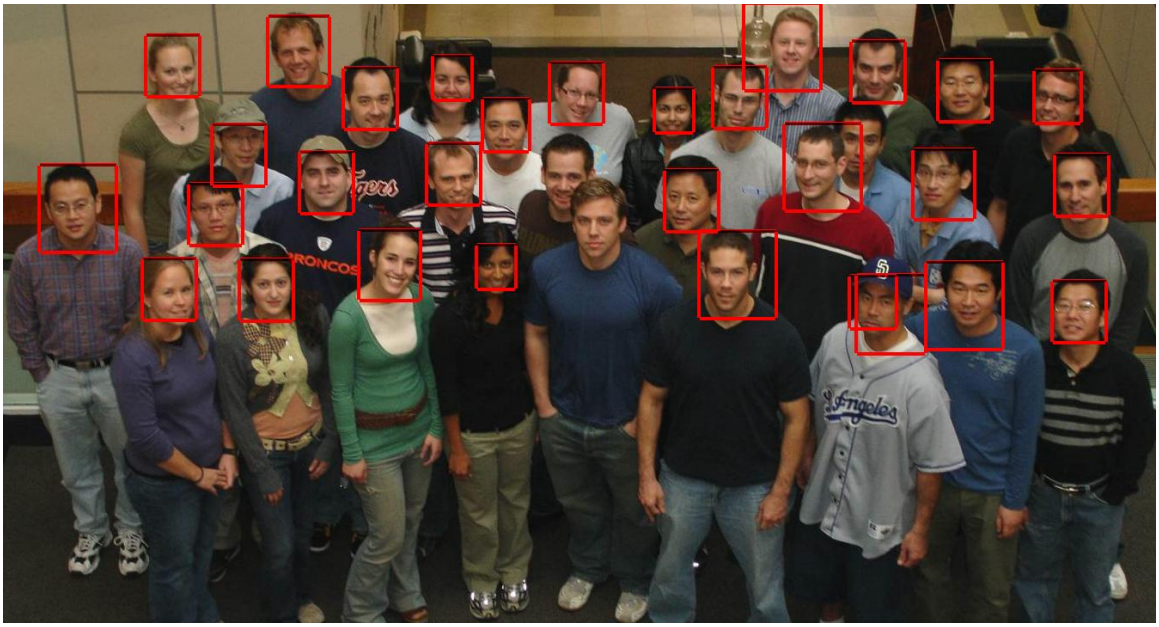

Figure 19: Final classified faces after SVM classification.

Table 1: Comparison of Face Detection Performance Using Different Eigen Features Along with HOG, Edge Ratio and Edge Statistics.

Feature	No. of Features	TP	TN	FP	FN	Precision	Detection Rate %	False Detection Rate %
Eigen	12	236	22	11	358	95.55	91.47	2.98
Normalized Eigen	12	236	22	9	360	96.33	91.47	2.44
Normalized Eigen	25	236	22	9	360	96.33	91.47	2.44

The terms in Table 1 are explained below.

- True positive (TP) is the number of faces that are detected from the algorithm.
- True negative (TN) is the number of faces that are not detected.
- False positive (FP) is the number of non-faces falsely detected as faces.
- False negative (FN) is the number of non-faces rejected from the classifier.
- Accuracy shows the proportion of true results, both true positives and true negatives, as computed as below

$$Accuracy = \frac{TP + TN}{TP + FP + FN + TN}$$

- Precision or positive prediction value is defined as the proportion of the true positives against all positive results. It is computed by

$$precision = \frac{TP}{TP + FP}$$

- Detection rate gives us a percentage of faces correctly detected.

$$DetectionRate = \frac{TP}{TotalFaces}$$

- False detection rate shows the percentage of non-faces falsely detected as face.

$$FalseDetectionRate = \frac{FP}{TotalNonFaces}$$

Table 1 clearly shows that the normalized features achieve better results. Since 25 normalized eigen features achieve the same performance as the 12 normalized eigen features, we use 12 normalized eigen features in our face detection.

Second, we ran a series of experiments to compare face detection performance using different feature combinations. In total, there are 15 possible combinations for the four kinds of features. However, four of these combinations cannot be used for SVM learning due to their skewed distribution. That is, support vectors cannot be computed for these four combinations. Table 2 summarizes face detection performance for the remaining 11 combinations of features. Here, ES represents edge statistics features, EIG represents normalized eigen features, and ER represents edge ratio features. Table 2 clearly show that using all four features gives us a good face detection rate and reduces false detection rate. Specifically, our face detection algorithm using four features on 56 images with 258 faces achieves a face detection rate of 91.47 and a false detection rate of 2.44%. Therefore, we consider all the four features in our further experiments.

Third, we ran a series of experiments to compare face detection performance using different constant values of *C* for the SVM training and classification. The results are shown below in Table 3. Clearly, the SVM classifier with *C* being 8 achieves good results with a face detection rate of 91.47% and a false detection rate of 2.17%.

Table 2: Comparison of Face Detection Performance for Different Features.

Feature	No. Of Features	TP	TN	FP	FN	Precision	Detection Rate %	False Detection Rate %
HOG	36	216	42	26	343	89.26	83.72	7.05
ES	5	2	256	0	369	100	0.77	0
HOG, EIG	48	215	43	27	342	88.84	83.33	7.32
HOG, ER	52	230	28	19	350	92.37	89.15	5.15
HOG, ES	41	223	35	20	349	91.77	86.43	5.42
EIG, ER	28	197	61	30	339	86.78	76.36	8.13
HOG, EIG, ER	64	233	25	12	357	95.1	90.31	3.25
HOG, ER, ES	57	233	25	16	353	93.57	90.31	4.34
HOG, EIG, ES	53	228	30	16	353	93.44	88.37	4.34
EIG, ER, ES	33	214	44	31	338	87.35	82.95	8.01
HOG, EIG, ER, ES	69	236	22	9	360	96.33	91.47	2.44

Table 3: Comparison of Face Detection Performance Using Different *C* Values for SVM Training and Classification.

C Value	TP	TN	FP	FN	Precision	Detection Rate %	False Detection Rate %
2^-1	235	23	8	361	96.71	91.08	2.17
2^1	236	22	9	360	96.33	91.47	2.44
2^3	236	22	8	361	96.72	91.47	2.17
2^5	236	22	9	360	96.33	91.47	2.44
2^7	236	22	9	360	96.33	91.47	2.44
Default value 1	236	22	9	360	96.33	91.47	2.44

In summary, our extensive experiments show the best configuration for our proposed face detection algorithm is to use all four complementary features (i.e., HOG features, normalized eigen features, edge ratio features, and edge statistics features) and set the *C* value for SVM as 2^3. This configuration achieves a high precision, while also maintaining a high detection rate and a low false detection rate.

Finally, we used Google's Picasa software to detect faces in our testing images. Five faces out of 258 faces were rejected, and 2 non-faces are detected as faces.

Our proposed system was implemented using Matlab 2011(b) on a computer with Intel Core 2 Duo 2.93 GHz Processor, having 4GB of memory. The running time to detect all faces in the three pictures as shown in Figures 15, 16, and 18 was 7.3, 25.1, and 7.6 seconds, respectively. Each of these input images are of size 1071 X 1500, 720 X 1342 and 400 X 600.

CHAPTER 5

CONCLUSIONS AND FUTURE WORK

We propose a novel face detection algorithm that seamlessly incorporates skin color modeling and thresholding, morphological enhancement, iterative template matching, feature extraction, and SVM classification to achieve decent face detection performance in a variety of images. Specifically, the proposed method is effective in detecting faces from typical photographs having multiple faces, single faces, faces wearing spectacles, faces of different races, faces with less light, and faces very close to each other. The experiments show that the proposed method outperforms the peer method [1], as our algorithm achieves a higher face detection rate and lower false detection rate. Specifically, our method improves the peer method by improving the face detection rate by 3.97% and reducing the false detection rate by 2.42%. Our experimental results on 258 faces using the best configuration show a face detection rate of 91.47% and a false detection rate of 2.17%. The major contributions of this project are:

- Employing the morphological enhancement operation to remove holes and separate a majority of adjacent objects.
- Designing an iterative template matching algorithm to locate multiple faces in a candidate patch.
- Designing two novel features, namely, an edge ratio feature and an edge statistics feature, to capture edge information in candidate patches.
- Combining the two novel features with two commonly used features, the HOG feature and the Eigen feature, to achieve optimal features for face detection.

Future work may include:

- Using more powerful and effective morphological enhancement to keep all the faces.
- Using different template matching approaches to reject a majority of the non-faces.
- Using other optimal features to represent the candidate patches.
- Using an AdaBoost classifier instead of SVM for the classification task.

REFERENCES

[1] Alajel, K.M., Xiang, W., and Leis, J. Face detection based on skin color modeling and modified Hausdorff distance. In *Proc. of IEEE Int. Conf. on Consumer Communications and Networking*, 2011, 399-404.

[2] Kim, I., Shim, J.H., and Yang, J. Face detection. EE368 Final Project, Stanford University, 2003, http://www.stanford.edu/class/ee368/Project_03/Project/reports/ee368group02.pdf

[3] Kherchaoui, S. and Houacine, A. Face detection based on a model of the skin color with constraints and template matching. In *Proc. of Int. Conf. on Machine and Web Intelligence*, 2010, 469-472.

[4] Martinez, A. and Benavente, R. The AR face database. CVC Technical Report #24, June 1998. http://www2.ece.ohio-state.edu/~aleix/ARdatabase

[5] Dalal, N. and Triggs, B. Histograms of oriented gradients for human detection. In *Proc. of IEEE Computer Society Conf. on Computer Vision and Pattern Recognition*, vol. I, 2005, 886-893.

[6] Lin, K-P. On the design and analysis of the privacy-preserving SVM classifier. *Proc. of IEEE Transactions on Knowledge and Data Engineering 23*, 11 (Nov. 2011), 1704-1717.

[7] Li, Z., Xue, L., and Tan, F. Face detection in complex background based on skin color features and improved AdaBoost algorithms,. In *Proc. of IEEE International Conference on Progress in Informatics and Computing (PIC)*, 2010, 723-727.

[8] Liang, L.H., Zhou, A.H., Xu, G.Y., Zhang, B., and Zhao, L.H. " survey of human face detection. *Chinese Journal of Computers 25*, 5 (2002), 450-458.

[9] Viola, P. and Jones, M. Rapid object detection using a boosted cascade of simple features. In *Proc. of IEEE Conference on Computer Vision and Pattern Recognition*, vol. I, 2001, 511-518.

[10] Chang, H. and Robles, U. Face detection. EE368: Final Project, Stanford University, May 2000, www-csstudents.stanford.edu

[11] Shavers, C. Li, R., and Lebby, G. "An SVM-based approach to face detection. In *Proc. of the 38th Southeastern Symposium in System Theory*, 2006, 362-366.

[12] Sha, F.S., Saul, L.K., and Lee, D.D. Multiplicative updates for nonnegative quadratic programming in support vector machines. *Advances in Neural Information Processing Systems 19*, 8 (2002), 2004-2031.

[13] Paul, P.P. and Gavrilova, M. "PCA-based geometric modeling for automatic face detection. In *Proc. of Int. Conf. on Computational Science and its Applications (ICCSA)*, 2011, 33-38.

[14] Jia,H-X. and Zhang, Y-J. Fast human face detection by boosting histogram of oriented Gradients. In *Proc. of Fourth International Conference on Image and Graphics*, 2007, 683-688.

[15] Jolliffe, I.T. *Principal Component Analysis*, 2nd Edition. Springer, 2002.

[16] Marius, D., Pennathur, S., and Rose, K. Face detection using color thresholding, and eigenimage template matching. EE368: Digital Image Processing Project, Stanford

University, May 2003. http://www.stanford.edu/class/ee368/Project_03/Project/reports/ee368group15.pdf

[17] Olivetti & Oracle Research Laboratory. ORL face database, AT & T Laboratories Cambridge http://www.cl.cam.ac.uk/research/dtg/attarchive/facedatabase.html

[18] Corinna Cortes and V. Vapnik, "Support-Vector Networks", Machine Learning, 20, 1995. http://www.springerlink.com/content/k238jx04hm87j80g/

[19] Cortes, C. and Vapnik, V. Support-vector networks. *Machine Learning 20*, (1995), http://www.springerlink.com/content/k238jx04hm87j80g/