4-1-2012

Face Detection from Images Using Support Vector Machine

Parin M. Shah

Face Detection from Images Using Support Vector Machine

A Thesis

Presented to

The Faculty of the Department of Computer Science

San José State University

In Partial Fulfillment

of the Requirements for the Degree

Master of Science

by

Parin M Shah

May 2012

SAN JOSE STATE UNIVERSITY

The Designated Thesis Committee Approves the Thesis Titled

Face Detection from Images

Using

Support Vector Machine

by

Parin M Shah

APPROVED FOR THE DEPARTMENT OF COMPUTER SCIENCE

SAN JOSÉ STATE UNIVERSITY

May 2012

Dr. Teng-Sheng Moh, Department of Computer Science	Date
Dr. Mark Stamp, Department of Computer Science	Date
Dr. Sami Khuri, Department of Computer Science	Date

ABSTRACT

Detection of patterns in images using classifiers is one of the most promising topics of research in the field of computer vision. A large number of practical applications for face detection exist and contemporary work even suggests that any specialized detectors can be approximated by using fast detection classifiers. In this project, I have developed an algorithm which will detect face from the input image with less false detection rate using combined effects of computer vision concepts. This algorithm utilizes the concept of recognizing skin color, detecting edges and extracting different features from face. The result is supported by the statistics obtained from calculating the parameters defining the parts of the face. The project also implements the highly powerful concept of Support Vector Machine that is used for the classification of images into face and non-face class. This classification is based on the training data set and indicators of luminance value, chrominance value, saturation value, elliptical value and nose, eye & mouth map values.

`

ACKNOWLEDGEMENTS

I am thankful to each and every person who has contributed his plethora for the completion of this project. Working on this project was unique experience. Knowledge and experience gained from this project will remain with me as an ingratiating memory.

I would like to express my sincere appreciation to my project advisor Dr. T.S. Moh for guidance, cooperation and time in making my project a success. Thanks for the benevolent support and kind attention. I would also like to thank my committee members Dr. Mark Stamp and Dr. Sami Khuri for their support and patience.

We would also like to thank our department for providing us with the necessary software required in our project. We are also thankful to the library for providing necessary books and materials required to learn different concepts for our project.

Last but not the least, sincere thanks to my parents, family and friends for constant moral support, inspiration, encouragement and blessings without which the task of completing the project successfully would have been next to impossible.

Table of Contents

1. Introduction……..11
2. Literature Review……14
   2.1. Skin color detection……14
   2.2. Edge detection……..15
   2.3. Extracting features of face……15
   2.4. Color space – RGB, YCbCr and HLS……17
   2.5. Support Vector Machine……..22
   2.6. Haar Cascade Classifier……30
   2.7. Programming in OpenCV……32
   2.8. Tools and Image data set……33
3. Algorithm……34
4. Results & Limitation……53
5. Future Work……..58
6. Conclusion……58
7. Reference……59

List of Figures

Figure 1. Color Cube image……18

Figure 2. HSV Color Model……20

Figure 3. Support Vector machine for linearly separable data……23

Figure 4. Maximum margin and support vectors for the given data sets……..24

Figure 5. Support Vector Machine for non-linearly separable data……..25

Figure 6. Haar Cascade Classifier……30

Figure 7. Input Image……35

Figure 8. RGB image converted to Binary image……36

Figure 9. YCbCr image converted to Binary image……38

Figure 10. HSV image converted to Binary image……40

Figure 11. Final Binary Image……42

Figure 12. Four major points in tracing and labeling points……45

Figure 13. Face Feature Extraction process……46

Figure 14. Final Result of my algorithm……50

Figure 15. Histogram calculation for RGB image ……52

Figure 16. Limitations-1 of my algorithm……..57

Figure 17. Limitations-2 of my algorithm……57

List of Equations

Equation 1. Calculate Normalized Red Component……..14

Equation 2. Calculate Normalized Green Component……..14

Equation 3. Eye map value for Chrominance Value……..16

Equation 4. Eye map value for Luminance Value……16

Equation 5. Mouth map value……..17

Equation 6. Formula for finding positive labels in SVM……23

Equation 7. Formula for finding negative labels in SVM……..23

Equation 8. Formula for finding hyper plane in SVM……23

Equation 9. Calculate Margin between plane of two class in SVM……..24

Equation 10. Soft Margin Hyper plane……..26

Equation 11. Polynomial Kernel……27

Equation 12. Radial Basis Function……27

Equation 13. Haar Classifier Features Equation……31

Equation 14. Blue Difference Chrominance Component……..38

Equation 15. Red Difference Chrominance Component……..39

Equation 16. Calculate Delta – Difference between minimum & maximum component……..40

Equation 17. Error Rate (ER)……53

Equation 18. Face Detection Success Rate (FDSR)……..54

List of Tables

Table 1. Results of our algorithm……54

Table 2. Results of AdaBoost Haar Cascade Detector Algorithm……55

1. Introduction

In the early few years several papers have been published on face detection in the community which discusses different technique like neural network, edge detectors and many more. There is a good survey by Chellapa, Wilson and Sirohey (1995) which tells about the trends of paper in face detection [11]. Previously, many researchers and engineers have designed different purpose specific and application specific detectors. The main goal of this kind of classifiers was to achieve a very high detection rate along with low computational cost. Few examples of different detectors are corner detectors, AdaBoost Haar Cascade detector by Jones and Viola [10] are very useful. This kind of detectors mostly use simple and fast classifiers that reject the most common negative samples and then they use progressively more complex classifiers to deal with the more difficult and odd negative samples.

Another approach to detect the face is through skin color classification algorithm [4]. Kjeldsen and Kender used the concept of different color space model to separate the skin patches from the image [18]. But if those algorithms are used solely, then it becomes difficult to detect more than one face from the image. Hsu, Abdel-Mottaleb, and Jain came up with the tactics of calculating the eye, mouth and nose map values [4]. Eyes, nose and mouth are significant features of face which distinguishes face from other

parts of the body. Studies have shown that they exhibit unique properties in the YCbCr color model, so it becomes easy to detect the face region using this feature extraction.

Poggio, Heisele and Ho presented a component based approach of locating facial components, extracting them and combining into a single feature vector which is used for classification of faces by Support Vector Machine. The authors used the gray scale image to define the feature vector for classification. Authors' using the approach of Support Vector Machine iterates through the whole image and compares it with face template to classify the region of interest. This takes very high computation time and error rate. Also, some of them can detect faces only from gray scale images. But our algorithm overcomes these drawbacks of classification. In our approach, preprocessing stages are applied which extracts the region of interest and feature vector is applied on this ROI instead of whole image for classification of the face region.

Detecting the face from color images poses various difficulties under varying lighting conditions, pose change and when there are additives on the face region like beard, moustache, etc. To overcome this, we applied certain preprocessing stages to my algorithm so that it detects the face region accurately with less error rate and low computational cost. In the first step, we would haul out and identify the face region based on the skin color segmentation algorithm. There are lots of variations in the human race or lighting condition, so to accurately detect the skin patch region we converted the image into RGB, YCbCr and HSV color space. This would take advantage of all models and find out the all the skin region from images. Afterwards the face feature

extraction process is applied which calculates the map values of different face features [4]. This threshold values and preprocessing values are passed to the feature vector of Linear SVM which would classify the image in face and non-face class. This classification is based on the training of the data set. Our data set consists of around 125 images with 305 faces. The images are taken from the internet and from my collection of photographs.

The next section discusses computer vision concepts used in this algorithm and approaches adopted by other authors. Section 3 describes the detailed approach and explanation of our algorithm. It gives step by step elucidation of how each step is executed in our algorithm. Section 4 discusses the results obtained from our algorithm and shows the comparison with Haar cascade classifier method. Section 5 points out the limitations of our algorithm and displays few scenario on which our algorithm does not show satisfactory results. In section 6 we listed out the future work which we will adopt in current methodology. Section 7 discusses the final conclusion and findings from our approach.

2. Literature Review

In this section we discuss various concepts like skin color detection, edge detection, morphological operators and support vector machine used by prominent authors of face detection.

2.1 Skin color detection

Many algorithms have been developed to determine the face using the skin color. There has been past findings to detect the skin color by determining the geometric correlation between position of hair and face [13]. In that, authors used the normalized RGB model to extract skin

$$r = \frac{\mathrm{R}}{R+G+B} \qquad \text{(eq. 1)}$$

$$g = \frac{\mathrm{G}}{R+G+B} \qquad \text{(eq. 2)}$$

For determining the skin color from the input images we have to convert the image into different color space like RGB, YCbCr and HLS. After wards they process each image to pull out the skin color utilizing the properties of color space. The implementation of this is very simple and depends upon the threshold value of the skin color. First step is to mask the skin color and then extract the part of the image which resembles the skin. The skin color detection has the limitation of detecting only one face per image. Crowley

and Coutaz [14] proposed simpler method for distinguishing pixels of skin by making use of skin color algorithm.

2.2 Edge detection

In the edge detection phase, edges of the object from the images are detected using a particular threshold value. Using the Gaussian filter the noise from the images is filtered out. The next step would involve calculating the intensity gradient of the image. This would help eliminate the varying lighting condition. Then the morphological operators like elation and dilation are applied which reduces the noise from the background image. The non-suppression is applied which helps in removing the points that are not part of the edge.

The most important step of this phase is to detect the edge of the objects. So depending upon the threshold pixel the edges are accepted or rejected. The value of the threshold is selected based on the average of the tested training data set of image.

2.3 Extracting features of face

Eyes, nose and mouth are the most prominent facial features which help in detecting the face. Based on the luminance and chrominance values we can locate the boundaries for eyes, nose and mouth from the image [2]. Luminance (Y) represents the brightness in the image i.e. the black and white portion of the image. It represents

the details of image without any of its color information whereas the chrominance represents the color information. Chrominance is represented as two color difference: where Cb is defined as Blue-Y' and Cr is defined as Red-Y'.

Eye Map

In this we calculate luminance component and chrominance component of eyes. The results of both these are added and combined. These values are calculated based on the properties of higher value of chrominance and lower value of luminance found near eyes [2]. The formula for calculating are follows

$$EyeMapC = \frac{1}{3}\left\{(C_b^2) + (\tilde{C}_r)^2 + (C_b/C_r)\right\} \quad \text{(eq. 3)}$$

$$EyeMapL = \frac{Y(x,y) \oplus g_\sigma(x,y)}{Y(x,y) \ominus g_\sigma(x,y) + 1} \quad \text{(eq. 4)}$$

Mouth Map

The mouth color has higher red component and lower blue component as compared to supplemental regions of the face. So we can conclude that the chrominance is greater than the luminance component. So we can calculate the mouth map as follows [2]:

$$MouthMap = C_r^2 \cdot \left(C_r^2 - \eta \cdot C_r / C_b\right)^2,$$
$$\eta = 0.95 \cdot \frac{\frac{1}{n}\sum_{(x,y)\in FG} C_r(x,y)^2}{\frac{1}{n}\sum_{(x,y)\in FG} C_r(x,y)/C_b(x,y)} \quad \text{(eq. 5)}$$

Using this we form the possible combinations for forming triangle of mouth and eye. From that we can detect whether the given image is face or not.

2.4 Color Space – RGB, YCbCr and HLS

Active development in the content based image field has led to great concentration to the study of skin color classification. Locating different image object like face, eye, car, etc can be exploited for different purpose like recognition, editing, detection, indexing and various other interactive purposes. Tracking the position of face using skin color also provides a vital stepping stone in studies related to facial expression [15]. Recent researches in the algorithms for face localization mostly prefer to utilize the color information to approximate the skin color. Skin color classification can be considered as an important task as most of the prevalent algorithm for face identification use color

information to estimate skin region. Estimating skin color region is most often considered as the vital step of face localization process. Nowadays, most of the research development in face detection using skin color is established on the concept of HSV, RGB and YCbCr color spaces. In this section we describe brief information about these different color models.

RGB

The perceived human color is proportional to the varying condition of illumination. Using the normalized color histogram we can sense the pixels for skin region which can be further normalized to resist the varying luminance condition. The Red, Green and Blue vector of the image is converted into a normalized form which provides a rapid means for detecting the skin and thereby confines the face region. There are limitations to this algorithm if there are some more skin regions like hands, neck, legs, etc in the image then it cannot detect it properly.

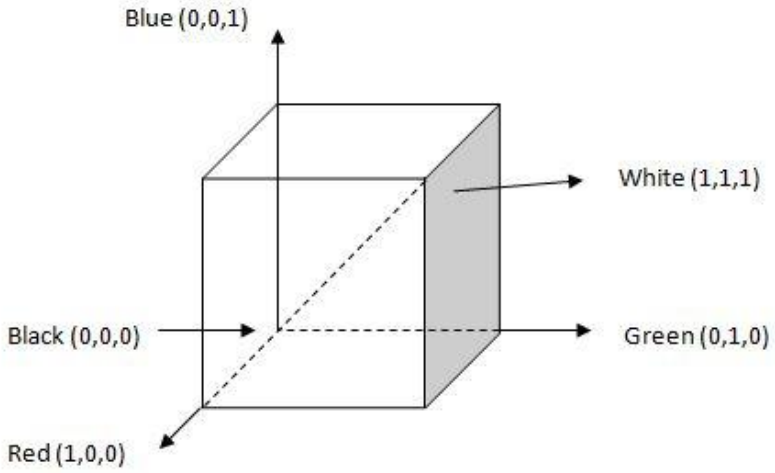

Fig 1. : Color Cube image.

YCbCr

Research related to the computer vision on YCbCr space have established that pixels that fits as the skin region display similar values for chrominance and luminance. Using chrominance and luminance values in skin color model can provide good result for detecting skin of different ethnicity and human races. Based upon the threshold value of that pixel, the face portion in color image is retrieved using the skin color distribution. This algorithm also has similar restriction that whole image should have only one face as the skin region.

YCbCr color space belongs to the family of video transmission color spaces that was designed specifically to handle the increasing demands for digital algorithms used in retrieving video related information. YCbCr was designed for digital color system whereas other color spaces that belonged to the same family like YUV and YIQ were designed for analog spaces [15]. All color models belonging to the family of video transmission color spaces separate Red, Green and Blue components into luminance and chrominance components that are useful in video compression applications. The specification of colors through this model space is less sensible but we follow the specification as defined by the Recommendation 601 [16]. The Recommendation 601 specifies 8 bit (i.e. 0 to 255) coding of YCbCr, whereby the luminance component Y has a digression of 219 and an offset of +16. This coding places black at code 16 and white at code 235, which allow for overshoot and undershoot [16]. The chrominance

components Cb and Cr have excursions of +112 and offset of +128, producing a range from 16 to 240 inclusively [16]. So, the Cb and Cr samples use the value 128 to encode a zero value, as used when encoding a white, grey or black area. The values used for sync encoding are 0 and 255.

HSV

We make use of the HSV space to separate the skin regions from background. Classification in HSV color space is the same as YCbCr color space but here the accountable values are saturation (S) and hue (H) instead of luminance and chrominance. A pixel is categorized to have skin tone if the value of Hue and Saturation fall within the specified threshold value and the distribution gives the localized face image [2].

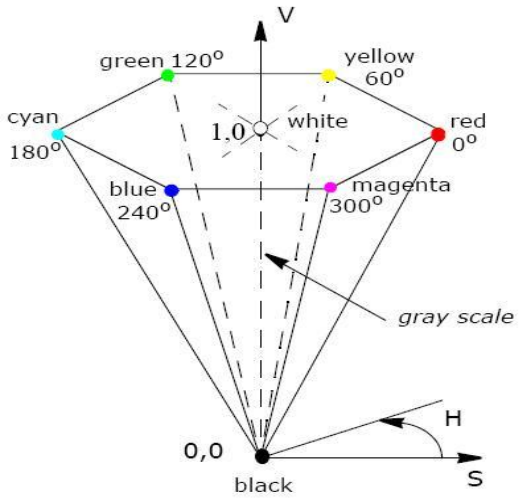

Fig 2. : HSV Color Model [16]

The HSV color space does not require an exact proportion of primary colors to get the resultant color; instead the variation is based on the values of Hue, Saturation and Intensity to produce a required color. For example an adjustment in saturation causes a shift between color violet and blue, a variation in intensity of pixel causes to make it darker or lighter. Operations like detecting object from image, histogram equalization, intensity transformation, etc are performed with much ease on an image in the HSV color space. Hue (H) is represented as the angle 0, varying from 0° to 360°, Saturation (S) corresponds to the radius, varying from 0 to 1 and Value (V) varies along the z axis with 0 being black and 1 being white.

In HSV color model, the color is of grey color with intensity equal to 1 if we have the corresponding value of pixel's Saturation =0. And if Saturation=0 then color is present on the boundaries of model represented by HSV. Moreover, color is more distant from grey, white or black if they have greater value of Saturation. Adjusting the hue will vary the color from red at 0°, through green at 120°, blue at 240°, and back to red at 360°. When Value = 0, black color is obtained and when Saturation = 0 the color of the image is grey or in different shades of grey. Hue value is indeterminate, when we have Saturation=0 and Value=0. By adjusting Value (Intensity), a color can be made darker or lighter. By maintaining Saturation= 1 and adjusting Value or Intensity we obtain different shades or traces of that particular color.

2.5 Support Vector Machine

In machine learning, task of deducing a category from supervised training data is known as Supervised Learning. In supervised learning the training data consist of a set of training examples, where each example is a pair consisting of an input and an anticipated output value. A supervised learning algorithm analyzes the training data and then predicts the correct output categorization for given data-set input. For e.g. Teacher teaches student to identify apple and oranges by giving some features of that. Next time when student sees apple or orange he can easily classify the object based on his learning from his teacher, this is called supervised learning. He can identify the object only if it is apple or orange, but if the given object was grapes the student cannot identify it.

Understanding Support Vector Machine for linearly separable data

Consider each image to be a single dot in the figure. And dot of different color specifies different category. Here we have image of two category and we have to find the boundary separating two images.

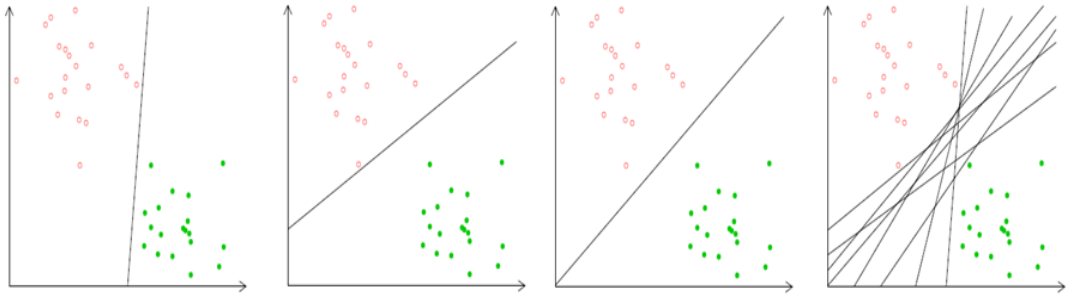

Fig 3. : Support Vector machine for linearly separable data.

The **Margin** of a linear classifier is the width by which the length of the boundary can be increased before hitting the data points of different category. The line is safe to pick having the highest margin between the two data-sets. The data points which lie on the margin are known as **Support Vectors**.

The next step is to find the hyper plane which best separates the two categories. Support Vector Machine performs this by taking a set of points and splitting them using different application specific mathematical formulas. From that we can find the positive and negative hyper plane. The mathematical formula for finding hyper plane is:

$$(\mathbf{p} \cdot q) + r = +1 \text{ (positive labels)} \qquad \text{(eq. 6)}$$

$$(p \cdot q) + r = -1 \text{ (negative labels)} \qquad \text{(eq. 7)}$$

$$(p \cdot q) + r = 0 \text{ (hyper plane)} \qquad \text{(eq. 8)}$$

From the equation above and using linear algebra we can find the values of p and r**.** Thus, we get the model that contains the answers for p and r and with margin value of $2/\sqrt{(k.k)}$. The margin is calculated as follow.

$$\text{Margin} = 2/\sqrt{(k.k)} \qquad \text{(eq. 9)}$$

In Support Vector Machine, this model is used to categorize new data. With the above solutions and calculated margin value, new coming data can be categorized into different category. The following figure demonstrates the margin and support vectors for linearly separable data.

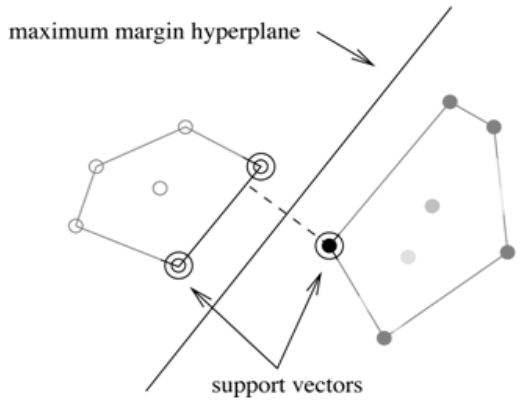

Fig 4. : Maximum margin and support vectors for the given data sets are shown in figure.

Understanding Support Vector Machine(SVM) for non-linearly separable data

For non-linearly separable plane, data are input in an input space which cannot be separated with a linear hyper plane. So, we map all the points to feature space using

specific type of kernel, in order to separate the non-linear data on a linear plane. After separating the points in the feature space we can map the points back to the input space with a curvy hyper plane. The following figure demonstrates the data flow of SVM.

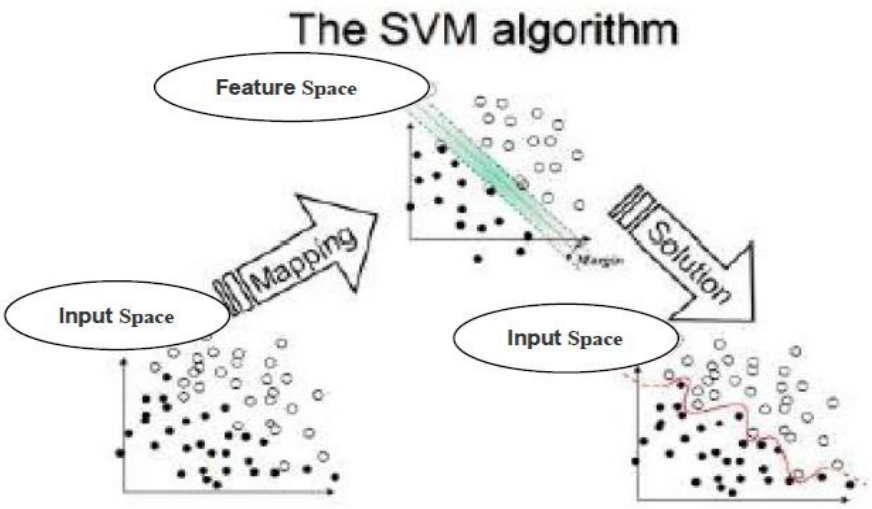

Fig 5. : Support Vector Machine for non-linearly separable data.

In reality, you will find that most of the data sets are not as simple and well behaved. There will be some points that are not correctly classified, these points that are far off from the classes, or points that are mixed together in a spiral or checkered pattern. Researchers have found the solution to tackle the problem of misclassification error through Support Vector Machine. It minimized the following equation to create what is called a **soft-margin hyper plane**.

$$\Phi(w,\xi) = \frac{1}{2}(w \cdot w) + C\left(\sum_{i=1}^{\ell}\xi_i\right)$$

s.t. $y_i(\langle \mathbf{w} \bullet \mathbf{x}_i \rangle - b) \geq 1 - \xi_i$
and $i = 1, 2, 3, \ldots \ell$

(eq. 10)

The higher value of C maximizes the margin value whereas the lower value of C lowers the margin value.

TYPES OF KERNEL

Computation of various points in the feature space can be very costly because feature space can be typically said to be infinite-dimensional. The kernel function is used for to reduce this cost. The reason is that the data points appear in dot product and the kernel function are able to compute the inner products of these points. So there is no need of mapping the points explicitly in feature space. By making use of different **kernel function** we can directly compute the data points through inner product and find equivalent points on the hyper plane.

The kernel functions which are being developed for SVM are still a research topic. No appropriate kernel has been found out which is universal for all kind of data. Anybody can develop their own kernel depending upon requirements.

The following are some basic types of kernel:

1.) Polynomial kernel with degree *d.*

$$K(\mathbf{x}, \mathbf{y}) = (\mathbf{x}^T\mathbf{y} + 1)^d \qquad \text{(eq.11)}$$

2.) Radial basis function kernel with width s

$$K(\mathbf{x}, \mathbf{y}) = \exp(-||\mathbf{x} - \mathbf{y}||^2/(2\sigma^2)) \qquad \text{(eq. 12)}$$

There are many other different kinds of kernel available. The user can also be developed kernel specific to the application.

STRENGTH OF KERNELS

Kernels are the most tricky and important part of using SVM because it creates the kernel matrix, which summarize all of the data points. Initially, a low degree polynomial kernel or Radial Basis Function (RBF) kernel with a reasonable width is a good try for most applications. Linear kernel is considered to be the most important choice for text classification because of the already-high-enough feature dimension. There are many ongoing researches to estimate the kernel matrix.

ADVANTAGES AND DISADVANTAGES OF SUPPORT VECTOR MACHINE

ADVANTAGES

There are many folds advantages of using the supervised learning approach of Support Vector Machine (SVM). They are very effective when we have very high dimensional spaces. Also, when number of dimensions becomes greater than the existing number of samples, in such cases too SVM is found to be very effective. SVM uses a subset of training point also known as support vectors to classify different objects hence it is memory efficient. Support Vector Machines are versatile, for different decision function we can define different kernel as long as they provide correct result. Depending upon our requirement and application we can choose types of kernel which is most productive for our application.

DISADVANTAGES

The disadvantage of SVM is that if the number of features is much greater than the number of samples, the method is likely to give poor performances. SVM gives efficient result for small training samples as compared to large ones. SVMs do not directly provide probability estimates, so these must be calculated using indirect techniques. Also, we can have Non-traditional data like strings and trees as input to SVM instead of featured vectors. Lastly selecting appropriate kernel for the project is a big issue which depends upon user's requirement.

Using SVM for Face Detection

The algorithm proposed by Osuna, Freund and Girosi detects faces by exhaustively scanning an image for face-like patterns at many possible scales, by dividing the original image into overlapping sub-images and classifying them using a SVM to determine the appropriate class [3]. Multiple scales are handled by examining windows taken from scaled versions of the original. Before storing the image some pre processing steps like masking, illumination and histogram equalization are performed. In the masking process unnecessary noise like the background pattern is reduced from the objects of interest. And then histogram equalization is used that manages the distribution of colors in images.

The images of class face and class non face are used to train the SVM using the kernel and upper bound margin values. Once a decision surface has been obtained through training, the run-time system is used over images that do not contain faces, and misclassifications are stored so they can be used as negative examples in subsequent training phases. In order to increase the precision of detecting face we can use negative examples for training misclassification class. There are ample non-face images available which can be trained in SVM. Non face images are richer and broader than face images.

2.6 Haar Cascade Classifier

In statistical model based training, we take multiple positive and negative samples and extract different features from these samples. These distinctive features are then compressed into statistical model parameters which are used as special property to classify different objects. By making adjustments in these parameters we can improve the accuracy of classification for these algorithms. The fundamental concept for detecting objects from images for Haar classifier is the Haar-like features. These features exploit the difference in contrast values between contiguously grouped pixels instead of using the intensity values of that particular pixel. These contrasting values between the grouped pixels are used to detect relative light and dark spot from the images. These two to three contiguous groups with a comparatively contrasting values form a Haar-like feature. In images we have object of different sizes, these Haar features can be scaled by increasing or decreasing the size of the grouped pixel being examined. This scaling of the pixels makes it possible to detect and extract objects with varying sizes.

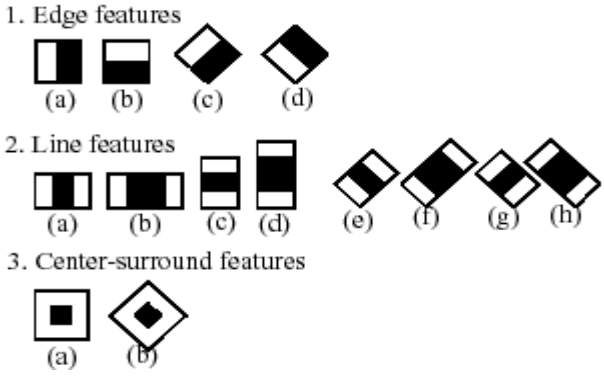

Figure 6. : Haar Cascade Classifier

Feature vector's value in Haar classifier is calculated as the difference between the sum of the pixels within the white and black region of interest. This region of interest is that region which iterates through the whole image to find out a specific template. This template is for the object that is to be detected from the image.

$$f_i = \mathrm{Sum}(\mathrm{r}_{\mathrm{i,\,white}}) - \mathrm{Sum}(\mathrm{r}_{\mathrm{i,\,black}}) \quad \text{.....eq (13)}$$

h(x) = 1 if f_i > threshold.

= -1 if f_i <threshold.

In opencv, we first take the positive and negative samples and form the corresponding database. After that we create a vector file and then build the classifier. After building the classifier we check for the performance and accuracy of the training dataset. If we find satisfying results we create a xml file which contains all the features for detecting objects. This is the whole procedure for detecting faces or any objects using Haar classifier. Afterwards we retrieve the classifier's data from the xml file and use this data to classify objects for the testing data set.

2.7 Programming in openCV

OpenCV acronym for Open Source Computer Vision Library is a library containing functions for computer vision. It is developed by Intel and now handled and supported by Willow Garage. The library is functional cross platform and runs on Windows, Android, FreeBSD, Maemo, iOS, OpenBSD, Linux and Mac OS. The current release of the library is obtained from the Sourceforge and they also provide the binaries for the user, so that they can develop according to their requirements. OpenCV makes use of CMake to compile source files to start using the library [12].

The main focus of this library is on the real-time image processing functionality and implementing the machine learning algorithms. By using this we can improve the cost of computation and take an initiative to advance the CPU – intensive applications. The areas of application where openCV can be useful are facial recognition system, mobile robotics, gesture recognition, segmentation, object identification, motion tracking and many more. OpenCv also includes a statistical machine learning library that supports the above areas of application. The name of the functions that supports this library are decision tree learning, expectation maximization, gradient boosting trees, , Naïve Bayes classifier, k- nearest neighbor, artificial neural network, support vector machine(SVM) and many more [12].

The goals of the developing libraries of opencv are manifold. I have utilized these goals to the benefit of my project:

- Advance vision research by providing open source as well as optimized code for basic vision infrastructure.

- Distribute vision knowledge by providing a common infrastructure for all developers so that they can share their research and knowledge.
- Making portable, performance-optimized code available for free, so that others can contribute computer vision based application.

The library was previously written in C language and because of this C interface the language is portable to different platforms. To increase the adoption of the openCV language, the wrapper class for different languages like C#, Python, Ruby and Java have been written [12].

2.8 Tools and Image dataset

The tools which I have used in my projects are Microsoft Visual C++and Git. The image dataset used in this project was developed from the images which I had clicked from my camera. Also some photos were taken from the internet (not containing copyright issues) to test varying conditions.

3. Algorithm

In this section we will discuss our algorithm which we have developed using different concepts as explained earlier.

High level design of my algorithm:

Input the image.

Convert the input image into different color space model with the goal of obtaining specific region from the image.

RGB space

YCbCr space

HLS space

Convert the image obtained from different color models into combined binary image. This step would put the value of pixel as 1 if it falls within the specified criteria; otherwise it puts the value as 0.

Filter the obtained binary image by applying different morphological operators like erosion and dilation.

Detect blob type region from the binary image and extract those elliptical region of interest.

Determine the face features from extracted region by calculating the mouth, eye and nose map.

Collect the parameters from above preprocessing stages in the form of support vectors and pass them to the function of linear SVM to classify them into face and non-face region.

Highlight the classified face region by using rectangle box in the original image.

Let's take an example for understanding how the face detection works. Following is the input image, which we are using to show various processing steps and display the corresponding result.

Fig 7. Input image

3.1 Convert Image into RGB, YCbCr and HSV color model and obtain Binary Image.

RGB Model

Red, Green and blue are the three additive primary colors for the RGB color space. Combination of these components in assorted proportion would produce different color. The RGB model is represented by a three-dimensional cube with red, green and blue defining the each axis (Figure 1). Black is defined at the origin (0, 0, 0) whereas white is at the other side the cube i.e. at (1, 1, 1). The gray color follows the diagonal line alongside the color cube starting from white to black. The image consists of three channels 8 bits each, with each color representing different channel. The RGB color model condenses the design of computer graphics systems, but is not ideal for all applications as all three components are highly correlated with each other, making it difficult to implement different image processing algorithms as they will be not resistant against different lighting conditions [15]. Here we analyze the image for skin region, so we will look for combination of R, G and B components in such a way that resultant color is the skin color.

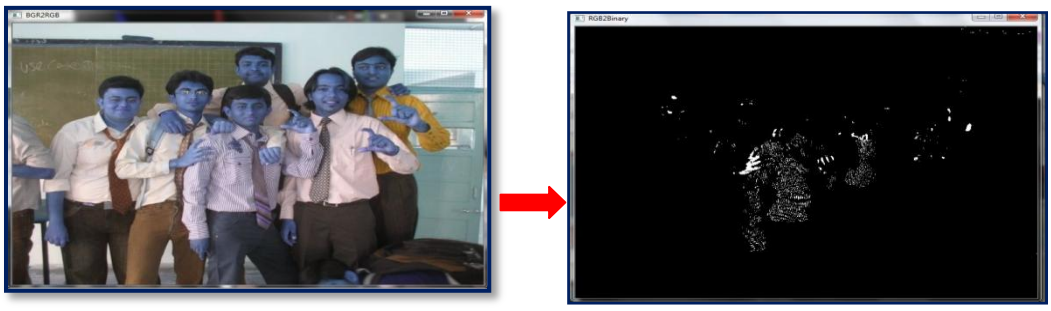

Image 8. : RGB image converted to Binary image.

The threshold values we use for extracting the skin region from the image are as follows:

Red Component: r_min = 194,r_max=230.

Blue Component: b_min=140,b_max = 168.

Green Component: g_min = 155,g_max = 184.

Pseudo – code for converting the RGB image into binary image.

```
// Iterate every pixel of the image.
For i:0 to number of pixels
        Then
                If(r_min<=r_value<=r_max  &&  b_min<=b_value<=b_max  &&
                g_min<=g_value<=g_max)
                                Pixel_value = 1;
                Else
                                Pixel_value = 0;
                End If.
End For.
```

YCbCr Model

Researchers have led to the results that skin region shows up same values of chrominance and luminance. So here, we implemented the skin color extraction algorithm that depends upon the chrominance and luminance values of the image. Skin region of humans vary depending upon the human race; experiments have showed that studies of skin region using Cb and Cr model have shown more successful results in detecting skin region. Minimum and maximum values are to be determined to classify a particular pixel as a skin tone. The range of this can be defined as Cr_min to Cr_max and *Cb_min to Cb_max.* This extraction would yield the different skin region present in the image.

Image 9. : YCbCr image converted to Binary image.

The formula by which we convert the RGB components into YCbCr components is given as follows:

cb_value = (-0.169 * Red) + (-0.331 * Green) + (0.5 * Blue) + 128; (eq. 14)

cr_value = (0.5 *Red) +(-0.419*Green) +(-0.081*Blue) + 128; (eq. 15)

And the threshold values we use for extracting the skin region are as follows:

cb_min = 112,cb_max = 151,cr_min = 139,cr_max = 155;

[cr1:cr2] = [139:155] and [cb1:cb2] = [112:151]

The above are the results obtained by analyzing the histogram of the faces present in the training dataset. After careful analyze, the average value of chrominance and luminance is taken which is equivalent to the above threshold range.

Pseudo – code for converting the YCbCr image into binary image.

```
// Iterate every pixel of the image.
For i:0 to number of pixels
    Then
        Calculate the Cr and Cb component from R,G & B using  eq no:
        If(crmin<=cr_value<=crmax && cbmin<=cb_value<=cbmax)
            Pixel_value = 1;
        Else
            Pixel_value = 0;
```

HSV Model

Skin color extraction in HSV model is similar to that of YCbCr model, but the accountable values here are Hue and Saturation. The HSV color space helps to distinguish the skin regions from background. Analogous to the YCbCr model the Hue values are chosen between h_min to h_max whereas the saturation values are taken from s_min to s_max. Depending upon this range, a pixel is classified as a skin tone.

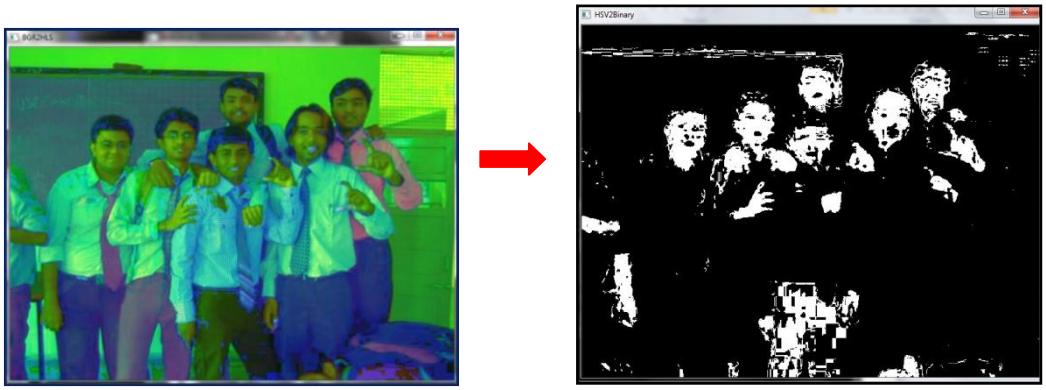

Image 10. : HSV image converted to Binary image.

Now, we will explain how the values of hue and saturation are calculated. First, calculate the minimum and maximum from the red, green and blue components of the image. The value defines the brightness of the color for that particular pixel and that is equal to the maximum component. Next step is to calculate the difference between minimum and maximum component.

$$Delta = max - min \qquad \text{(eq. 16)}$$

The saturation value is defined as delta over max. And the threshold values we use for extracting the skin region are as follows:

h_min = 6 and h_max = 18; [h1:h2] = [6:18]

The threshold values are obtained by observing the histogram of the faces present in the training dataset. After careful analyze, the average value of hue was taken as in the range of 6 to 18.

Pseudo – code for converting the HSV image into binary image.

```
// Iterate every pixel of the image.
For i:0 to number of pixels
    Then
        Calculate min and max component from R,G & B. Also calculate
        delta = max-min.
        Calculate Brightness value = max and Saturation = delta/max.
        Calculate Hue.
        Wrap the outlier points.
            If(h_value > 6 && h_value <= 18)
                Pixel_value = 1;
            Else
                Pixel_value = 0;
```

Now I have created a final binary image, by adding all three binary images created from different color model. We then applied basic morphological operators like erosion and dilation to the binary image, which removes the noise from the image.

Fig 11. : Final Binary Image

The structuring element is the fundamental concept of morphological operators. It is defined as an arrangement of pixels (defining shape) on which an origin is located.

Applying a morphological filter consists of probing each pixel of the image using this structuring element. When the origin of the structuring element is aligned

with a given pixel, its intersection with the image defines a set of pixels on which a particular morphological operation is applied. In principle, the structuring element can be of any shape, but most often, a simple shape such as a square, circle, or diamond with the origin at the center is used (mainly for efficiency reasons)

As with all other morphological filters, the two filters of this recipe operate on the set of pixels (or neighborhood) around each pixel, as defined by the structuring element. Recall that when applied to a given pixel, the anchor point of the structuring element is aligned with this pixel location, and all pixels intersecting the structuring element are included in the current set. Erosion replaces the current pixel with the minimum pixel value found in the defined pixel set. Dilation is the complementary operator, and it replaces the current pixel with the maximum pixel value found in the defined pixel set. Since the input binary image contains only black (0) and white (255) pixels, each pixel is replaced by either a white or black pixel. A good way to picture the effect of these two operators is to think in terms of background (black) and foreground (white) objects. With erosion, if the structuring element when placed at a given pixel location touches the background (that is, one of the pixels in the intersecting set is black), then this pixel will be sent to background. While in the case of dilation, if the structuring element on a background pixel touches a foreground object, then this pixel will be assigned a white value. This explains

why in the eroded image, the size of the objects has been reduced. Similarly, the dilated objects are now larger and some of the "holes" inside of them have been filled.

During the erosion and dilation operation when an empty matrix is passed in the parameters it takes a 3x3 square (taken as cv::Mat()) as a structuring element. Depending upon your requirement and application you can vary the size of the structuring element by imparting a non-zero element matrix that defines the anchor point.

3.2 Detecting Blobs from Binary Image

After getting the binary image, the next step is to obtain the blobs that may represent the face. Blobs are typically elliptical in shape or can be considered as a connected dense component.

In this method of contour detecting and tracking, it scans a binary image from top to bottom and from left to right for every line scanned. Conceptually, this process of contour detection can be divided into four stages as illustrated in Figures a to d. In stage 1 represented by figure a, when an external contour point is encountered the first time, we make a complete trace of the contour till we again reach to the original point. In this way of tracking contour points, we label all the points of that contour. In the next Figure b, when a labeled external contour point A' is encountered, we track all the succeeding black colored pixels (if they are present) through the scan line and assign them the same tag as A'. In

Figure c, when an internal counter point, say B, is encountered the first time, we assign B the same label as the external contour of the same component. We then trace the internal contour containing B and also assign to all contour points the same label as B. In Figure d, when a labeled internal contour point, say B', is encountered, we follow the scan line to find all subsequent black pixels (if they exist) and assign them the same label as B'.

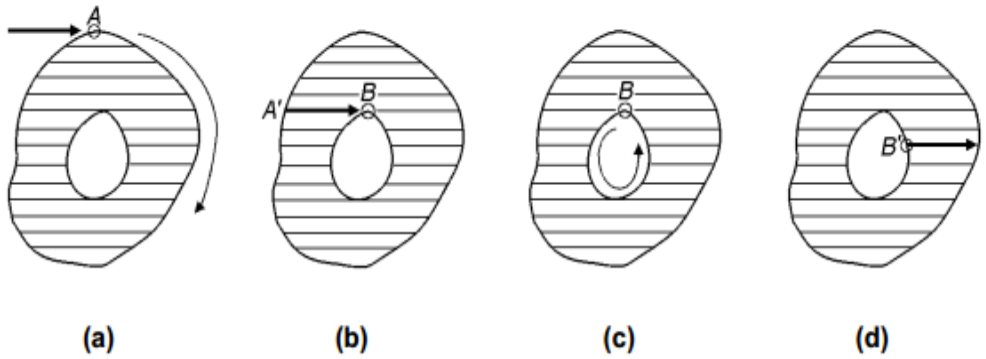

Figure 12. : Four major steps in tracing and labeling points

We used the cvBlobsLib which is based on the above concept and contains various functions to detect the blob. The result of this stage is store in elements of CvSeq which contains array for location of detected blobs. The locations of these blobs are stored in the form of cvPoints which contains the starting co-ordinates and height and width of that particular contour.

3.3 Face Feature Extraction

Next preprocessing stage involves extracting different face features like eyes, mouth and nose from the detected blobs. Once the blobs are detected the

extracted face region becomes the Region of Interest (ROI), and is converted into YCbCr model space. This image is then split into three different channels of chrominance and luminance. Then using eq. (3), eq. (4) and eq. (5) we calculate the eye map values and mouth map values. Then we will combine these images and apply the filtering operation of dilation and erosion [4]. This will yield the eye and mouth region.

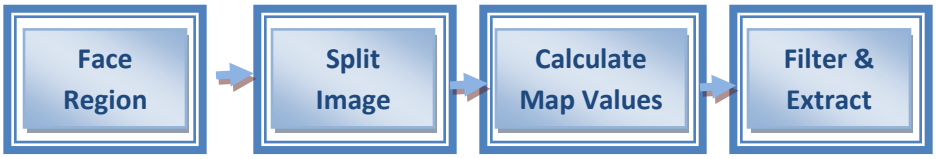

Fig 13. : Face Feature Extraction process

After locating the eyes and mouth region the geometrical position of the spots are checked. If the eyes and mouth center form a structure that is equivalent to the equilateral triangle, then we can confirm the position of face. The locations of the center of these regions are passed over to the next stage for classification, which checks over with face template from the data set.

3.4 Classification of ROI using Support Vector Machine

There is a large difference between the face and other parts of the body containing skin. Features like rich texture, eyes, mouth create a huge difference between the face and other body parts like hands, arms, shoulders, legs and neck. The Linear SVM kernel function $K(xi, xj) = \langle xi , xj \rangle$ is designed to classify the

data into face and non-face and we used openCV library to train the samples. As the number and size of faces are often different in images, in order to detect all the faces, we have to take pyramid analysis. In that case we scale every image few times till the size of the skin color region is fixed. In every scale, we scan the original color image from left to right, top to bottom in the effective skin color regions, and intercept image as a detected sample. After the detection sample is processed with a mask, the sample feature vector is put into the SVM classifiers to classify. This classifier uses the data from the pre processing stages.

In our algorithm, values obtained from the preprocessing stages of the training data sets are formed into vector and used as 2D array with class label of face and non-face. The class label and training data are stored as follows: for eg

Float labels[2]= {1,1}

Float trainingdata[2][2]={vector of pixel values for RGB, YCbCr or HSV color model, map values}.

The result of this classifier is then cascaded to the face feature extraction. In this phase of classification the face features extraction. In this phase the different map values like nose map, eye map and mouth map are calculated for the face region and then passed into the array of training data with corresponding label of face class. Afterwards we require the training data to be stored as an objecto of Mats, so we create Mat object from the array of floats:

Mat trainingDatatoMat(3, 2, CV_32FC1, trainingData);

Mat labelstoMat (3, 1, CV_32FC1, labels);

Now before starting the training we need to define certain parameters for SVM like which kind of kernel to use, type of SVM to use and termination condition.

CvSVMParams parameter;

Parameter.svm_type = CvSVM::C_SVC; (used for n-class classification & n>=2)

Parameter.kernel_type=CvSVM::LINEAR; (Select different types of kernel)

Parameter.term_crit =cvTermCriteria(CV_TERMCRIT_ITER, 100, 1e-6);

(specifying maximum number of iterations and tolerance error before algorithm ends)

Next step is to train the SVM using the training data. For that we use the following function:

CvSVM SVM_train;

SVM_train.train(trainingdatatoMat, labelstoMat, Mat(), Mat(), parameter);

Once we train the data, then we use the predict method to determine face or non-face class. We can also get the useful information about the support vector. Cv::get_support_vector gives general information about the support vectors

used and cv::get_support_Vector_count gives the number of support vectors used.

There will be times when the same face will be detected many times in the adjacent position or at different scales. We need to merge the overlap regions, and obtain the region's location and size. In most cases we have tried to remove this overlapping region by neglecting the region which are similar in area greater than 70% and considering them as single face. This can be the reason when certain face is not detected when they are nearby each other. According to the average of the location and size, we determine the location and size of the face candidate regions. The next step is passing the value of mouth region and eye region to the classifier. If the centers of the region form geometrical shape then we can conclude that the region of interest is the face region; otherwise its non-face region. This will classify the region and will mark the region's height and width, which will help to highlight the face by drawing a bounded rectangle.

Now, here we can mark the difference in how the Support Vector Machine is implemented. Previously many researchers used SVM by iterating over the whole image and finding respective face from defined classifier. As it has to go through entire image the computation time and cost is high. Our approach is different, in which we apply the preprocessing stage and extract the supposed face region. This region of interest passes through the SVM and classifies into face and non-face. Many authors used the preprocessing approach, but they

used for gray-scale images only. Whereas our algorithm overcomes all drawback and works for different color images.

3.5 Final Results

Here we can observe the final result of the image. All the faces in the input image are detected. The results and limitation of the algorithm are discussed in the next section. After the SVM classifies the region into face label, we highlight that part of the original image bounded with rectangle of red color.

Fig 14. : Final Result of my algorithm

3.6 Analyzing the Histogram of each channel

In the previous sections, we found the average values of different channels determining the skin region by analyzing the histogram. The obvious way of

doing this is to split the image into different channels, then calculate histograms of array of images and then normalize this array. Starting from the basics, we define histogram as number of counts of data belonging to a particular range. For eg: Consider the following matrix which has different values of pixel ranging from 0 to 255 for single channel.

23	47	87	23	87	47	78	87	71
53	134	64	53	64	134	173	64	145
222	210	42	222	42	210	140	42	119
213	190	67	213	67	190	90	67	188
45	178	96	45	96	178	19	96	10
90	167	213	90	213	167	240	213	60
88	2	34	88	34	2	80	34	51

Now to count the number of values of pixel we can segment the values in particular range known as bin. So, values ranging from 0 to 15 comes under bin1, values from 16 to 31 comes under bin2 and so on. Considering this we have total of 16 bin each having values range of 16. Now to figure out the actual average value related to skin region we extracted the face from the images in the training dataset and obtained corresponding histogram for each channel. Then we observed the different bins and channel with maximum values, those values

were taken as threshold for that particular color model. This process was repeated for all color models like RGB, YCbCr and HSV.

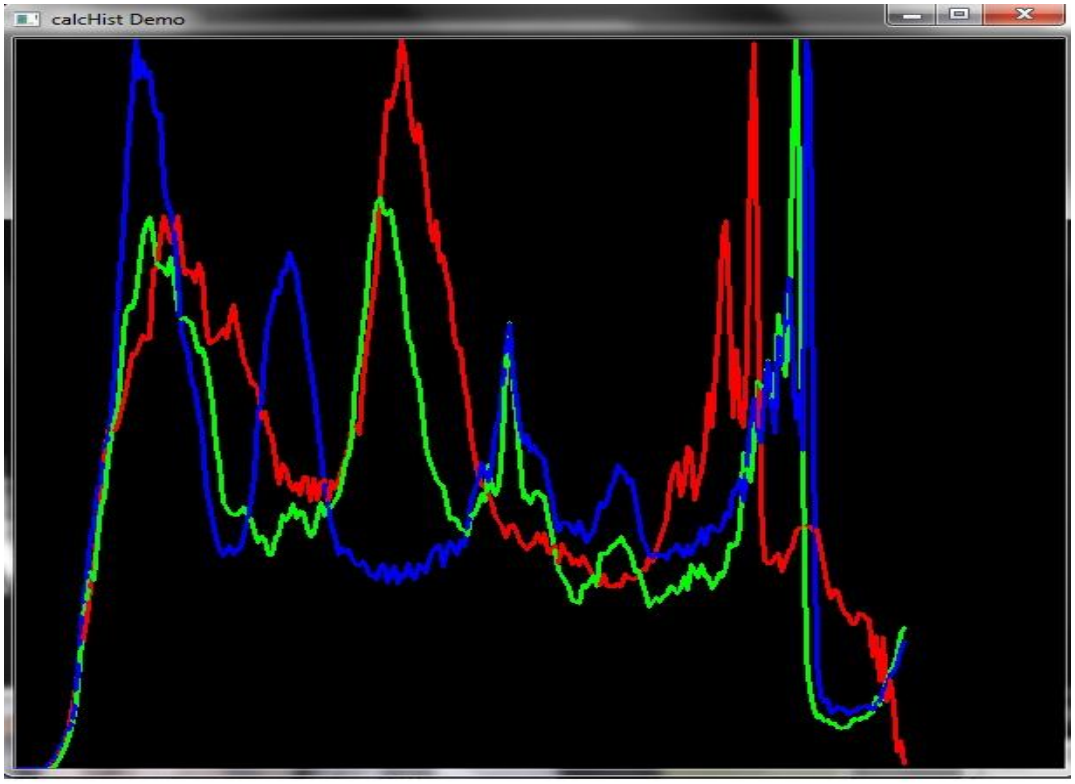

Figure 15. Histogram calculation for RGB image.

We observed each and every image from the training data set and calculated the histogram values for every pixel corresponding to the face region. Thus we showed how to calculate the threshold values pertaining to the skin region in the images.

4. Results and Limitations

Result

The proposed algorithm was trained and evaluated on the dataset of around 125 images containing 305 face images. This dataset was build from my collection of photographs and some random images from internet. The test images consisted of images with different lumination condition – night time, daytime and combination of them. The image formats acceptable to the algorithm are jpeg, png, bmp, etc. The dataset consist of images of size ranging from 400x320 to 2000x1800. If the size of the image is more than 2000x1800 then it would create problem in processing the image. We implemented the algorithm on an Intel® Core™ i5 CPU M 430 @ 2.27 GHz with 4.00 GB of memory.

We defined two parameters to measure the success of our proposed algorithm. First is the Error Rate (ER) which is defined as number of false detection in the image divide by the total number of detections (face and non-face).

$$\text{Error Rate (ER)} = \frac{\text{Number of false detection}}{\text{Total Number of Detections}} \times 100\% \qquad \text{(eq .17)}$$

Here, number of false detection means those objects that are identified as face but are not face. The total number of detections is the summation of face detected and non-face object detected.

Second parameter that is of interest is the Face Detection Success Rate (FDSR). It is defined as the number of faces detected correctly over the total number of faces.

Face Detection Success Rate (FDSR) = $\frac{\text{Number of face detected}}{\text{Total Number of Faces}}$ x 100% (eq. 18)

The experimental results of my face detecting algorithm are described in the below table. The algorithm achieved a FDSR of 90.82% and ER of 16.57%.

Total Number of Faces	**Number of faces detected**	**Number of non faces detected**	**Total Number of Detection**
305	277	55	305

Table 1. : Results of my algorithm.

Using equation (17) to calculate ER we get,

Error Rate (ER) = $\frac{55}{277+55}$ x 100% = **16.57%**

Using equation (18) to calculate FDSR we get,

Face Detection Success Rate (FDSR) = $\frac{277}{305}$ x 100% = **90.82%**

Our proposed scheme was compared with the AdaBoost classifier used by Viola and Jones [10]. We tested the same training data set in this method and found

comparatively better results. The algorithm achieved a FDSR of 84.92% and ER of 25.79%.

Total Number of Faces	Number of faces detected	Number of non faces detected	Total Number of Detection
305	259	90	305

Table 2. : Results of Adaboost Detector algorithm.

Using equation (17) to calculate ER we get,

Error Rate (ER) = $\frac{90}{259+90}$ X 100% = **25.79%**

Using equation () to calculate FDSR we get,

Face Detection Success Rate (FDSR) = $\frac{255}{305}$ x 100% = **84.92%**

From results conducted on both methods, we can observe that the error rate or false detection rate is comparatively low in our algorithm. The reason for this is that we perform extra preprocessing step of filtering noise using morphological operator. Also we calculate the nose, eye and mouth map values describing the face feature extraction process, which helps in reducing the error rate.

Limitation

There are certain limitations on my proposed algorithm for face detection. First of all, I have carefully analyzed my data set and based on that I have kept the threshold ranges for detecting skin region. If due to varying condition, the skin color does not fall into specified range than my preprocessing stage will not be able to detect the skin region. Sufficient cautionary steps are taken to detect the skin region by using three different color models, but there may be times when skin region is not detected.

Second, if the face in the image is tilted by some angle then it cannot detect the face because we consider the image height and width during certain calculation like merging, overlapping. If the face is tilted then the height and width are changes, so the supposed face region is automatically neglected in the pre processing stage.

Also, my algorithm calculates map values during the face feature extraction process. This is highly dependent on the visibility of face features. For example in the below figure the person in the middle has covered his eyes, in that case the face is not detected.

Figure 16. : Limitation-1 of Algorithm

There can be variation to the lighting condition because of which the skin color is not detected in the final binary image, in such case we are not able to detect the faces from the image. Below figure represents the scenario.

Figure 17. : Limitation-2 of algorithm

5. Future Work

In depth research for detecting the variation in human pose should be carried out. Current algorithm needs some efforts in detecting various pose from images. Also, experiments should be carried out to observe and analyze different kernels for classification using Support Vector Machine.

6. Conclusion

In this project I have presented face detection algorithm using the skin color detection, edge detection, facial feature extraction and using the concept of different color space. After these pre processing stages, the algorithm utilizes the highly powerful concept of Support Vector Machine (SVM) to classify the image into face and non-face region. We have significantly reduced the misclassification errors as compared to the Adaboost classifier of Viola and Jones [10]. The computation time for our algorithm is very less and the accuracy on the image data set of 125 images with 305 face image is around 90% with error rate of approximate 16%. We overcame the limitation of detecting one face from image using skin color algorithm; by combining the concept of different color space and face feature extraction process.

7. Reference

[1] Garcia, C., & Tziritas, G. (1999). Face detection using quantized skin color regions merging and wavelet packet analysis. Multimedia, IEEE Transactions on, 1(3), 264-277.

[2] Hsu, R. L., Abdel-Mottaleb, M., & Jain, A. K. (2002). Face detection in color images. Pattern Analysis and Machine Intelligence, IEEE Transactions on, 24(5), 696-706.

[3] Osuna, E., Freund, R., & Girosi, F. (1997). Training support vector machines: An application to face detection. Paper presented at the Cvpr, 130.

[4] Rein-Lien Hsu, Abdel-Mottaleb, M., & Jain, A. K. (2002). Face detection in color images. Pattern Analysis and Machine Intelligence, IEEE Transactions on, 24(5), 696-706.

[5] Rowley, H. A., Baluja, S., & Kanade, T. (1998). Neural network-based face detection. Pattern Analysis and Machine Intelligence, IEEE Transactions on, 20(1), 23-38.

[6] VIOLA, P. AND JONES, M.J. 2004. Robust real-time face detection. International journal of computer vision 57, 137-154. .

[7] Yang, G., & Huang, T. S. (1994). Human face detection in a complex background. Pattern Recognition, 27(1), 53-63.

[8] Shih, P., & Liu, C. (2006). Face detection using discriminating feature analysis and support vector machine.

[9] KADIR, T., BRADY, M. 2001. Saliency, Scale and Image Description. International Journal of Computer Vision, Volume 45, Number 2.

[10] P. Viola and M. Jones.(2001). Rapid Object Detection Using a Boosted Cascade of Simple Features. IEEE Conf. CVPR, Vol. 1, pp. 511-518.

[11] R. Chellapa, C. Wilson, and S. Sirohey. (1995). Human and machine recognition of faces: a survey. Proceedings of the IEEE, 83(5):705-741.

[12] OpenCV library function. Retrived October 12, 2011 from http://en.wikipedia.org/wiki/OpenCV.

[13] Yao-Jiunn Chen, & Yen-Chun Lin. (2007). Simple face-detection algorithm based on minimum facial features. Paper presented at the Industrial Electronics. Society, 2007. IECON 2007. 3rd Annual Conference of the IEEE, 455-460.

[14] Crowley, J. L. and Coutaz, J. (1997). "Vision for Man Machine Interaction," Robotics and Autonomous Systems, Vol. 19, pp. 347-358.

[15] Sanjay Kr. Singh, D. S. Chauhan, Mayank Vatsa and Richa Singh. (2003). A Robust Skin Color Based Face Detection Algorithm.Tamkang Journal of Science and Engineering, Vol. 6, No. 4, pp. 227-234.

[16] Recommendation 601. Retrieved February 12, 2012 from http://en.wikipedia.org/wiki/Rec. 601.

[17] HSV Color Model. Retrieved December 25th, 2011 from

http://software.intel.com/sites/products/documentation/hpc/ipp/ippi/ippi_ch6/ch6_color_models.htm.

[18] Kjeldsen, R. and Kender., J. (1996). “Finding Skin in Color Images,” Proceedings of the Second international Conference on Automatic Face and Gesture Recognition, pp. 312-317.

[19] C. C. Chang, C.J. Lin. (2008).LIBSVM -- A Library for Support Vector Machines. http://www.csie.ntu.edu.tw/~cjlin/libsvm/