Spring 5-22-2017

Cascaded Facial Detection Algorithms To Improve Recognition

Edmund Yee
San Jose State University

Cascaded Facial Detection Algorithms To Improve Recognition

A Thesis
Presented to
The Faculty of the Department of Computer Science
San Jose State University

In Partial Fulfillment
of the Requirements for the Degree
Master of Science

By
Edmund Yee
May 2017

The Designated Thesis Committee Approves the Thesis Titled

Cascaded Facial Detection Algorithms To Improve Recognition

by
Edmund Yee

APPROVED FOR THE DEPARTMENTS OF COMPUTER SCIENCE

SAN JOSE STATE UNIVERSITY

May 2017

Dr. Robert Chun	Department of Computer Science
Dr. Thomas Austin	Department of Computer Science
Dr. Melody Moh	Department of Computer Science

Abstract

The desire to be able to use computer programs to recognize certain biometric qualities of people have been desired by several different types of organizations. One of these qualities worked on and has achieved moderate success is facial detection and recognition. Being able to use computers to determine where and who a face is has generated several different algorithms to solve this problem with different benefits and drawbacks. At the backbone of each algorithm is the desire for it to be quick and accurate. By cascading face detection algorithms, accuracy can be improved but runtime will subsequently be increased. Neural networks, once trained, have the ability to quickly categorize objects and assign them identifiers. Combining cascaded face detectors and neural networks, a face in an image can be detected and recognized. In this paper, three different types of facial detection algorithms are combined in various configurations to test the accuracy of face detection at the cost of runtime. By feeding these faces into a convolution neural network, we can begin identifying who the person is.

Acknowledgments

I would like to thank Dr. Robert Chun for his continued support and providing me the guidance necessary to work on this project. I would like to thank my advisor Dr. Robert Chun and committee members Dr. Thomas Austin and Dr. Melody Moh for teaching me core skills needed to succeed and reviewing my project. And finally I would like to thank my parents for their patience and advice they gave me throughout my life.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

1 Introduction

The ability to use technology to recognize biometric characteristics of human beings have grown continuously. The desire to be able to programmatically represent a human being has become a hot topic in academia as different algorithms are introduced to accurately and quickly identify a human being. Human features such as the face, hand, fingers, eyes, and voice have been heavily researched to develop algorithms to find the best features to represent and identify them.

One of the most identifying features of human beings is the face. People can instinctively recognize other human beings by facial features and comparing it against the people they remember. Despite human beings using the face to recognize other humans, quick and accurate methods in computers for identifying a person have proven challenging and have not gained traction until recent years. Even with the rapid progress made, much research still needs to be done to improve the accuracy compared to a human being's ability to recognize faces. A subset of image recognition problems, face detection and identification problems generally follow these four steps:

1. Determine if there is a face in the picture.
2. Normalize the picture by doing things like correcting orientation, removing noise, and adjusting for lightning conditions.
3. Identify the features of the face to use. Generally key points of the face are used so the whole face is not needed.
4. Compare the face's extracted features against a database of other faces.

Facial recognition uses two different performance parameters to determine its usefulness. The first parameter is the correct detection rate. This parameter states how many faces were correctly detected and associated with the right person. The second parameter is the false detection rate. This parameter states how many images were incorrectly identified as a face, incorrectly associated with a known person, or both.

This paper aims to implement one method of face recognition. Cascading face detection algorithms has shown to have accuracy improvements in certain configurations [3]. This paper will cascade three different face detection algorithms in multiple configurations to see which cascade yields the best results. The three face detection algorithms are Histograms of Oriented Gradients (HOG), Viola-Jones object detection, and skin segmentation detection to determine if there is a face in the picture. If a face has been determined to be inside the picture, then the face will be extracted from the image and fed into a neural network to recognize the face. To be able to associate a face with a name, the neural network must be trained beforehand with a set of known images. This is similar to how humans must be introduced to a person and look at their face first before they can begin to recognize it as someone they know. A computer is similar in the sense it needs to scan the image for a face and then begin to extract facial features like eyes, nose, ears, or hair and try to guess who this person is based on all the people it has previously seen.

This paper provides an introduction to face detection and recognition technologies and proposes an enhancement on conventional methods of facial recognition to help improve accuracy. Section 2 details challenges and issues with face detection and recognition. Section 3 details technologies related to face detection. Section 4 describes how after recognizing a face is present in an image, what facial recognition technologies are and how they associate a face with an identifier. Section 5 covers various papers that have done research into face detection, facial recognition, or both. Section 6 lists this paper's proposed enhancement in face detection and facial recognition. Section 7 covers technologies and tools used in this project. Section 8 details the implementation of the project. Section 9 shows the results associated with this paper's proposed enhancement and compares it against other technologies. Section 10 provides this paper's conclusion. Finally, section 11 covers possible future work and ideas.

Portions of the research in this paper use the FERET database of facial images collected under the FERET program, sponsored by the DOD Counterdrug Technology Development Program Office.

2 Challenges With Face Detection And Recognition

Despite the many advancements in the field of face detection and recognition, there are still many challenges faced. The ability to utilize computers to recognize people have drawn interest from many groups ranging from law enforcement to industry. The algorithm must not only be accurate and fast, but it must be affordable as well.

Differences in illumination of the scene, changes in pose, orientation, and expression can cause many algorithms to classify a face incorrectly [4]. Clever tricks attempt to solve the problems like orientation attempt to warp the image so the faces are always facing the same direction [5]. Then points of the face like the nose or eyes can be extracted as features and compared to a database of known people who match those features. However these methods do not always guarantee accuracy since it has to do dubious actions like guess the other side of the face before the algorithm can even begin to extract features from it. Therefore many algorithms work best when a full frontal face is given rather than say the profile of a human face.

3 Face Detection

Face detection is a complex problem computers have had problems with solving. Humans have the innate ability to easily identify other human beings by their faces in real time. Computers on the other hand have not been so fortunate. The issue with computers is they must be given exact instructions on how to process an image and look for objects like faces. Subtle differences like if a person is smiling or frowning can cause a computer to misidentify a face. Other changes like offsets, orientation of object, or discoloration in images can cause computers to categorize an image incorrectly from one they had previously seen if simple matching algorithms are used. What is needed is a generic method to grab what is most common in all human faces and look only at those points.

An algorithm called the Viola-Jones object detection framework [6] provides an efficient and quick way to detect faces in images, though the algorithm can also be applied to many different types of objects. The algorithm achieves its efficiency by taking advantage of certain qualities of an image. It employs several tricks to quickly identify a face by looking for certain features that are indicative of a face being present and failing quickly when critical features for a face are not present in the image.

Face detection is a subset of object classification problems. Another way to detect objects, particularly parts of the human body, is skin segmentation. Skin segmentation takes advantage of the skin coloration in human beings. Skin segmentation works by looking at different color spaces other than RGB due to clustering issues with RGB. Different color spaces cluster the human skin color tones better. Thresholds are then set for each color space and this filter is applied to the image. The resultant image is then filtered again to look for blobs in the region and discarding smaller blobs. Another filter is used to determine which regions could potentially be a face so things like hands are removed and faces are the only remaining objects.

Another object classification solution is HOG. HOG relies on edge detection of shapes. This concept notes how edges will have gradient changes around edges of shapes. Blocks of the image is gathered into condensed features which are represented by gradient histograms. These features are then trained to determine which ones best represent the face. Future images are then fed through the system and scored to determine if a face exists.

3.1 Viola-Jones Object Detection Framework

The Viola-Jones object detection framework is able to quickly and accurately detect faces in images. Despite the resourcefulness of the Viola-Jones algorithm, there are certain drawbacks. To improve speed, the images fed into the algorithm are normalized into grayscale. This generally isn't a problem since the primary goal is the detection of faces and not technically recognizing who they belong to. The second drawback is the original algorithm makes no attempt at faces in different orientations. This means if only a person's profile is in the picture, the algorithm would not be able to detect it as a human face. Another classification targeting just profile faces would have to be built to detect these faces. Despite these drawbacks, the algorithm is still very popular for detecting faces despite its invention since 2001. It accomplishes face detection by performing the following three steps:

1. Detect areas of interest using the Haar basis functions
2. Adaptive Boosting (AdaBoost) training to determine relevant features
3. Cascading classifiers

3.1.1 Haar Basis Functions

The algorithm uses a method similar to the Haar basis function to determine areas of interest without observing every single pixel. The Haar basis function uses rectangles divided between black and white and overlays these rectangles onto the image. These various rectangles are seen in Figure 1.

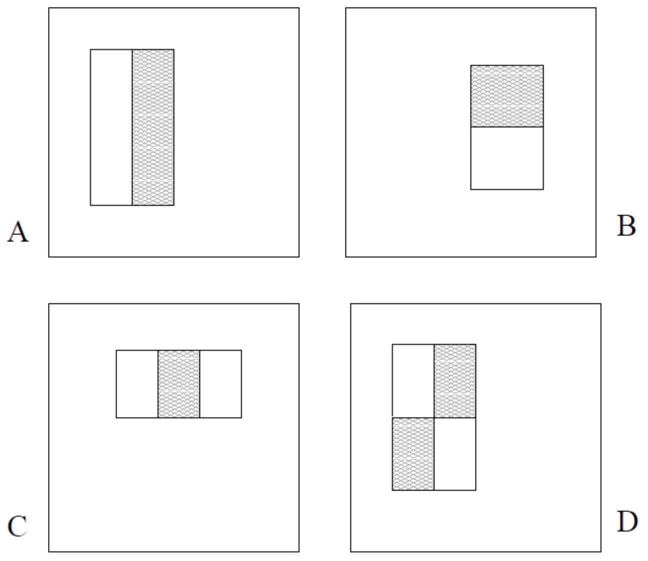


Figure 1: Figure (A) and (B) shows a two-rectangle features. Figure (C) shows a three-rectangle feature. Figure (D) shows a four-rectangle feature. [6]

The rectangles used help divide the image into regions of interest. Each rectangle from Figure 1 is applied to each pixel on the image. What ends up happening is for certain pixels, the rectangles dark and white regions align with the picture's dark and white regions. At first, the smallest rectangle possible is applied to each pixel and gradually grows bigger. For example, the beginning rectangle for Figure 1.A would be a 1x1 rectangle. The next iteration would be a slightly larger rectangle with size 2x2. What ends up happening is certain rectangles and certain sizes begin to overlap on some important features of a human face. For example, the rectangles can begin to separate certain parts of the human face. The rectangle in Figure 1.B when expanded can overlay on top of a person's eyes and cheeks. This feature observes the cheeks of a person are lighter than the eyes. This allows the algorithm to guess that if the image has this rectangle feature in the image, it is possible that it refers to a person's eyes and cheeks.

The Viola-Jones algorithm divides the picture in 24x24 pixel windows and each rectangle feature at every size up to the window size is applied to the all the windows. Given three rectangle features and the 24x24 window size, there was 45,396 unique features that had to be examined in the original paper [6]. Each of the colored rectangles in Figure 1 is given a value, say +1 for the white rectangle region and -1 for the black rectangle region. The goal is to get a singular value to represent the rectangle feature. This can be done by using Equation 1.

$$Value \ = \ \sum(pixels\ in\ black\ area) - \sum(pixels\ in\ white\ area). \tag{1}$$

For example, the rectangle in Figure 1.A would look like Table 1 for 1x1 and 2x2 rectangle.

Table 1: Haar Basis Rectangle Example Values

+1	-1

(a) 1x1 Rectangle

+1	-1
+1	-1

(b) 2x2 Rectangle

As stated earlier, the summation of the pixels in the white and black regions are necessary to get a singular value to represent the rectangular feature. However computing the sums for all the pixels for every single rectangular feature as it is applied to the entire image can be time consuming. Since speed is an important factor, Viola-Jones employs a math trick called the integral image that speeds up this process dramatically.

The basic idea behind the integral image is done by calculating all the pixel values above and to the left of the interested pixel point. For example, Figure 2 shows an example of a rectangle split up into four different regions with four points on the edges of the rectangle D. Every point in the image needs to be recalculated according to the integral image. An example for reassigning the values in points 1 and 4 in Figure 2 can be see in Equations 2 and 3.

$$Point\ 1\ Value = \sum(sum\ of\ pixels\ in\ rectangle\ A) \tag{2}$$

$$Point\ 4\ Value = \sum(sum\ of\ pixels\ in\ rectangle\ A,\ B,\ C,\ and\ D) \tag{3}$$

What this allows is the quick calculation of the total pixel sum of any region by simply adding the points diagonal to each other and subtracting the two. For example, to calculate the sum of pixels for rectangle D, Equation 4 can be used.

$$D = (Point\ 1 + Point\ 4) - (Point\ 2 + Point\ 3) \tag{4}$$

This provides a constant time calculation of the sum of pixels by calculating beforehand the integral image.

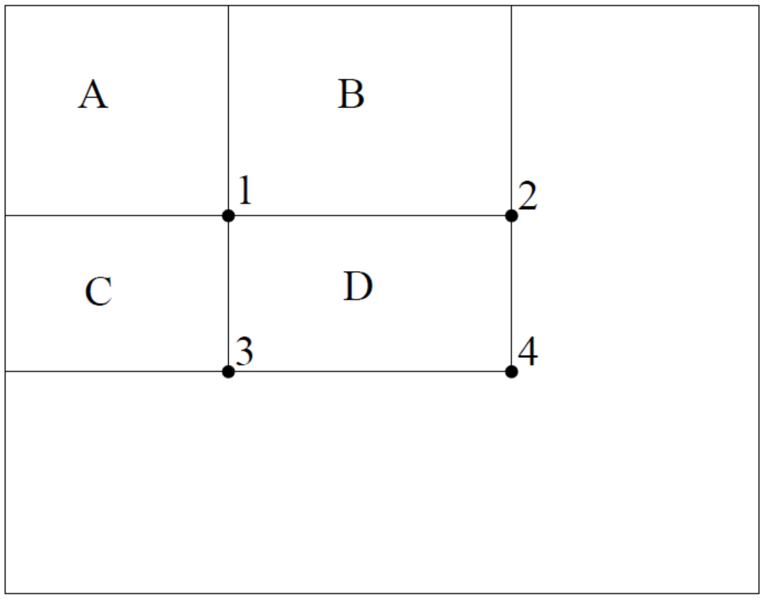


Figure 2: Integral image where the sum of all the pixels can easily be calculated by knowing the edges. [6]

3.1.2 Adaptive Boosting (Adaboost)

As noted before, there are 45,396 unique features that needs to be explored for every single image. Even with the speedup provided by the integral image formula, this is a computationally heavy task to perform. AdaBoost is the other algorithm chosen to help significantly boost performance by the authors of Viola-Jones to further improve the performance.

AdaBoost is a machine learning algorithm that takes several weak classifiers and combines them into a weighted sum to form a strong classifier. The goal of AdaBoost is to be trained with enough examples that good features are associated with heavy weight values and bad features are given low weight values. For a learned classifier to be effective, three conditions must be met [7]:

1. Trained on enough training examples.
2. Provide a good fit to those training examples (low training error).
3. Should have simple rules.

Eventually with enough training examples, a formula takes shape for each feature with an associated weight that can predict new images that it has never seen before. What this allows is the removal of certain features with low weights to improve the speed of the face detection algorithm. Since not all the rectangular features are necessarily important, say a 1x1 pixel rectangle on an image with large faces would not really say anything significant. However a much larger version of this rectangular feature can begin to identify a person's eyes and cheeks. Initial experiments by Viola-Jones have shown that by reducing the number of features to just 200, they were able to achieve a success rate of 95% and a false positive rate of 1 in 14,084 on their testing data set. More features can be kept to increase the accuracy rate at additional cost of computational time.

3.1.3 Cascading Classifiers

The final speedup suggested by the Viola-Jones algorithm is to use cascaded classifiers. What this means is the formula generated by AdaBoost can be split up into certain classes that each represent an important part of the human face. For example, one class could contain features that identify the eyes. Another class could be features that identify a person's cheeks. By splitting up the classes, a smaller amount of computation needs to be done at each class. By chaining these classes together, the same formula would be achieved had it not been split up. An extra piece of logic however is inserted between the classes that stops further computations if the previous classifier did not pass. This fail fast technique saves computational cycles by not requiring further calculations if say the algorithm could not detect eyes or a nose in the picture. An example two stage classifier can be seen in Figure 3. As seen in the figure, the input is immediately discarded if it fails the first stage and continues on to be evaluated by the next stage until no more stages are available.

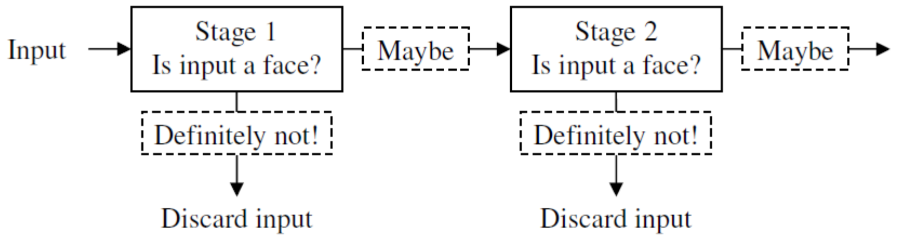


Figure 3: Example Two Stage Cascaded Classifier. [8]

3.2 Skin Segmentation

Another face detection scheme is using the skin color of humans to determine if there is a face in the image. One tricky aspect of this technology is the human skin color varies between people. Various colors like blackish, yellowish, brownish, and whitish come up when observing human skin color. In addition, skin color can appear different due to different lighting conditions in the image. In the world of computing, one of the most common ways to represent an image is with RGB, otherwise known as the red (R), green (G), or blue (B) color wheel. By increasing or decreasing each of the color components, it is possible to create any other color like yellow or purple by adjusting the red, green, or blue values accordingly. Each color is given 8 bits, so one pixel in an image is 24 bits.

There are also other color representations like HSV or YCbCr. These two color representations differ slightly from RGB because they represent color with hue or luminescence. Through experimentation, it has been found that RGB is generally not ideal for color based representation or color analysis since it mixes color intensity information together and nonuniform characteristics [9]. Since images can have different lighting conditions, it is useful to separate intensity information out which results in better clustering for color analysis.

HSV stands for hue (H), saturation (S), and value (V). YCbCr stands for luma (Y), chroma blue (Cb), and chroma red (Cr). By adjusting each of these values like RGB, it is possible to create any color. These color spaces differ from RGB but have formulas that can translate between them all. The formula converting RGB to HSV can be seen in Figure 4 and the formula converting RGB to YCbCr can be seen in Figure 5.

$$H = arccos \frac{\frac{1}{2}(2R-G-B)}{\sqrt{(R-G)^2-(R-B)(G-B)}}$$

$$S = \frac{max(R,G,B)-min(R,G,B)}{max(R,G,B)}$$

$$V = max(R,G,B)$$

Figure 4: Formula converting RGB to HSV. [9]

$$\begin{bmatrix} Y \\ Cb \\ Cr \end{bmatrix} = \begin{bmatrix} 16 \\ 128 \\ 128 \end{bmatrix} + \begin{bmatrix} 0.279 & 0.504 & 0.098 \\ -0.148 & -0.291 & 0.439 \\ 0.439 & -0.368 & -0.071 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix}$$

Figure 5: Formula converting RGB to YCbCr. [9]

As seen in Figure 4 and 5, the conversion process from RGB to HSV is more complex than the conversion process from RGB to YCbCr. Compared to HSV, YCbCr has an efficient separation of color and intensity information. The separation of intensity from an image is useful because of luminance in an image. Luminance creates problems in color analysis because it can vary across a person's face due to lighting conditions in the image. Therefore for image analysis purposes, it is useful to remove this factor in images and compare only the chrominance values.

The general idea behind skin segmentation is converting an RGB image to a different color space like HSV or YCbCr. A database of different skin color tones from various ethnicities is taken to create thresholds. Unlike RGB, other color spaces will begin to cluster around certain values to determine where the thresholds should be. The threshold mask, like $75 < \text{Cb} < 250$ and $10 < \text{Cr} < 100$ $\text{Y} > 80$, is applied to every pixel in the image effectively creating a mask. What is3 left is an image with non-skin regions appearing dark and possible skin regions like the face or hands are highlighted in the image as seen in Figure 6.

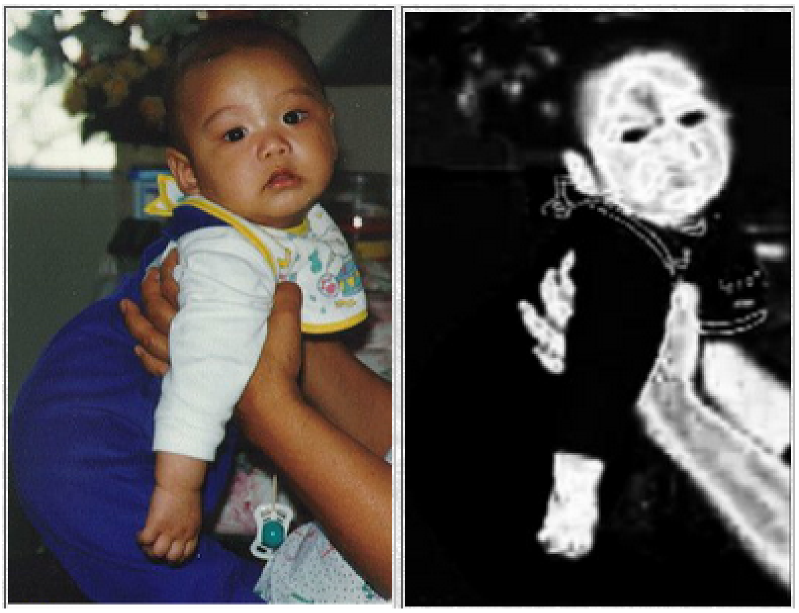


Figure 6: Skin Segmentation Filtered Image. [10]

Once the highlighted regions are observed, the next step is to determine regions of the highlighted areas. The next approach is to set thresholds for what a human face should have. Further thresholds can be applied including how many holes the region has, the width, or the height. Once all the thresholds have been satisfied, the algorithm will determine whether the region specified is a face or not.

3.3 Histogram of Oriented Gradients

HOG is an object classification algorithm that gained popularity for detecting humans after the paper using HOG for pedestrian detection by Navneet Dalal and Bill Triggs [11] was published. HOG focuses on using the edges of objects to detect shapes. Like other object detection algorithms, features are extracted from the image to begin understanding what components make up a face. A HOG feature descriptor is a distribution of histograms which consist of directions of oriented gradients used as features [12]. Gradients of an image are useful because the magnitude of gradients are large around edges and corners as intensity abruptly changes.

HOG is split between a training step and an evaluation step. The training step is similar to the evaluation step but is used to generate several HOG descriptors to train a classifier. The training step needs to be performed on positive and negative images containing content the user wants to identify. In this paper's case, the positive set should contain a set of faces and the negative a set of non-faces. Once a sufficiently trained classifier is created, the system can begin to evaluate future images using the following process:

1. Preprocess image.
2. Compute gradients.
3. Distribute gradients into histograms.
4. Block normalization.
5. HOG feature descriptor creation.
6. Use a classifier to determine similarity with pre-trained model.

3.3.1 Preprocess Image

To first step is to process the image. This step includes various tasks like resizing the image so its size ratio is 1:2 pixels. For example, the size chosen by Dalal and Triggs was 64x128 pixels. In addition, the image can be normalized by applying gamma correction to the image to deal with luminance issues. However this provides minimal amount of improvements and can be skipped.

3.3.2 Compute Gradients

The next step is apply a one dimensional kernel across the image as seen in Table 2. For color images, each channel's gradient is computed and the largest value is taken as the pixel's gradient vector. The magnitudes for x and y can be represented by Equations 5 and 6.

Table 2: Gradient Computation Kernel

-1	0	1

(a) HOG Horizontal Kernel

-1
0
1

(b) HOG Vertical Kernel

$$g_x = P(y, x+1) - P(y, x-1) \tag{5}$$

$$g_y = P(y+1, x) - P(y-1, x) \tag{6}$$

Figure 7 shows the two filters applied to a frontal facing image. The edges of the face can still be clearly seen while unimportant information like the background color are no longer visible.

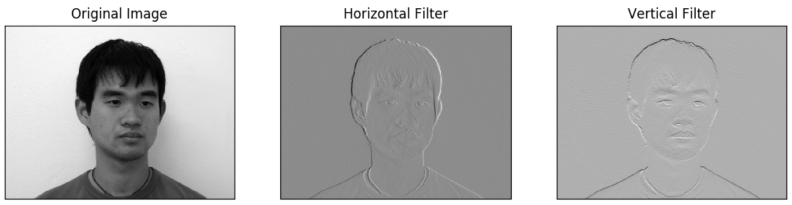


Figure 7: HOG Horizontal and Vertical Filter Applied.

Once the kernel has been applied to the image, the pixel at that point has a vertical and horizontal gradient value. The magnitude (g) and angle (θ) of that pixel can be represented by Equations 7 and 8:

$$g = \sqrt{g_x^2 + g_y^2} \tag{7}$$

$$\theta = \arctan(g_y/g_x) \tag{8}$$

3.3.3 Distribute Gradients Into Histograms

The gradients by themselves can be further grouped together to reduce dimensionality. The image is divided into 16x16 pixel blocks. Insides each block is an 8x8 pixels cell where the cell histogram will be calculated. Each block overlaps with one another so the same cell will influence other blocks. Cell histograms consist of 9 bins corresponding to the angle range 0 to 180 degrees where each bin is incremented by 20 degrees. For angles in between two bins, the pixel's magnitude value is proportionally split between the two bins. An example of the histogram can be seen in Figure 8.

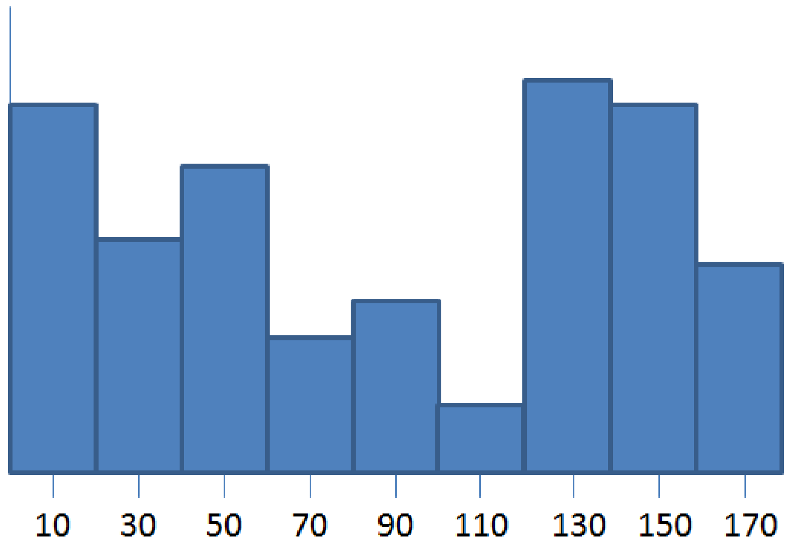


Figure 8: HOG 9 Bin Histogram. [13]

Each block is then condensed into four sets of histograms representing the gradient magnitudes and gradient orientations.

3.3.4 Block Normalization

Images are susceptible to illumination changes. Making an image darker by decreasing all the pixel values or making an image brighter by increasing all the pixel values will affect the histogram values. The descriptor needs to be able to be independent of lighting variations. There are various normalization algorithms used, but the end result is pixel value changes will not affect the feature vector representing the block. The normalization is done on the block's histogram value. Normalizing the block's histogram values will achieve the same affect as normalizing the pixel values individually and collecting the histogram of those normalized.

3.3.5 HOG Feature Creation

HOG feature descriptors are used to describe the image in an accurate and efficient manner. The descriptor is a concatenated vector of all the cell block values. For the default 64x128 pixels image size used by Navneet Dalal and Bill Triggs, the picture would be divided into 7 horizontal blocks and 15 vertical blocks assuming a stride step of 8 pixels and block size of 16x16 pixels. The final feature descriptor would then be a vector defined by the equation: 7 horizontal blocks * 15 vertical blocks * 4 cells per block * 9 histogram bins = 3780 dimensional vectors.

3.3.6 Classifier

The next step is to feed this dimensional vector into a classifier to determine which features in the vector best indicate where the face is. Training images using the methods mentioned above are fed through this system with positive and negative images. Eventually the system learns to discern the important parts. Future images can then be fed through to locate faces in the image.

4 Face Recognition

Coupled closely together with face detection is face recognition. The first step to start with in face recognition is face detection. It is necessary to determine where the face is in the image before any extraction of features can be done for recognition. Another criteria is to build a database of existing faces to begin training which type of features are associated with a person. Then when an unknown face comes in without any labels, the algorithm can begin to extract features and compare it against faces it had been previously trained with. Feature extraction can be classified into two different groups: local and global features. Local features individually identifies parts of the face like the nose, eyes, and cheeks by itself. Not all features have to match to associate the face with a person. This is typically more robust since it allows for small amount of changes to the face, i.e. scars, and still correctly recognize the person. Global features takes the whole image and converts them into a set of features. This is generally less robust because it doesn't allow for many changes in the face to occur. However global features are generally less discriminate and are more lenient towards certain features in a face being different to achieve high recognition. These features are fed into a neural network for further processing and identification.

4.1 Neural Networks

To be able to classify faces, a direct comparison is not practical because faces can have different orientations, sizes, or offsets. Neural networks are very robust in handling these changes but require training sets of images before it can correctly identify objects. Neural networks have been used in several different types of problems because of their strengths in classifying objects by recognizing patterns and their ability to cluster these patterns. The basis of neural networks is inspired by the human brain and how it recognizes objects. Inside the human brain, there are a set of neurons, or nodes, that are all interconnected together. These nodes each contain some logic that determines whether or not the input should be classified a certain way. The node then passes along its result to the next node until it reaches to the output node which determines the final result based on all the processing done by the nodes before it. As seen in Figure 9, there is first a set of input nodes, otherwise known as the input layer, at the far left where the initial features are extracted. These features are further fed into the network and fed into one or more hidden layers in the middle that further processes the features and assigns weights. Finally the results of the hidden layers are fed into the output layer at the far right.

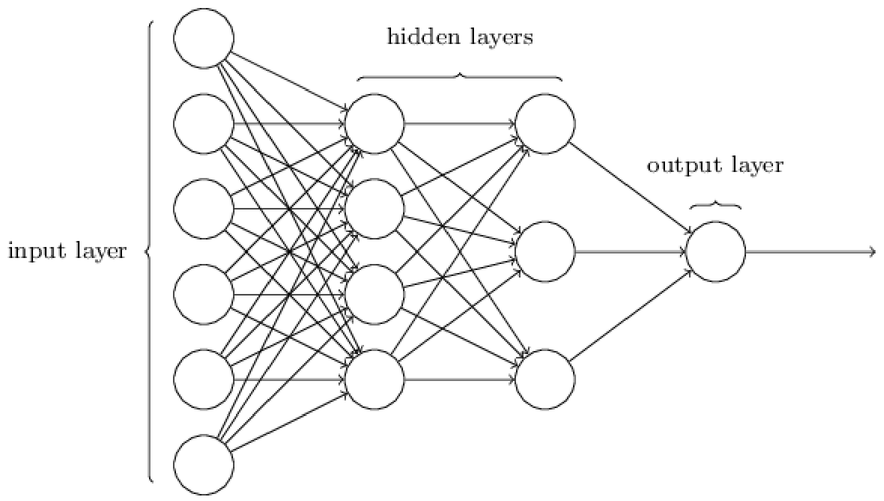


Figure 9: Artificial Neural Network example. [14]

Behind the scenes there is a mathematical formula that is slowly being generated from the features represented as weights. Ideally, the goal is to have as many weights as possible since it means there are more details that are analyzed. For example, when classifying an apple, one feature can be the color of the object. However since apples can have multiple colors depending on the type of apple they are, i.e. green or red, more features need to be extracted from the object to better identify whether or not the object is an apple.

To be a useful neural network, it must be accurate in its predictions. This is represented by a quadratic formula known as the cost function. As the network is being trained by the training set, this formula is continuously being adjusted to adjust the weights to achieve the best values associated with the weights to produce the most accurate result. In other words, the minimum needs to be found to find the best accuracy to associate the weights with. To find the minimum, calculus can be used to find the derivative of the equation. Visually, the cost formula can be graphed in a multidimensional graph as seen in Figure 10. This graph has only two features, but is applicable to more features which will introduce further dimensions. The variable v_1 and v_2 represent the weights and C represents the cost.

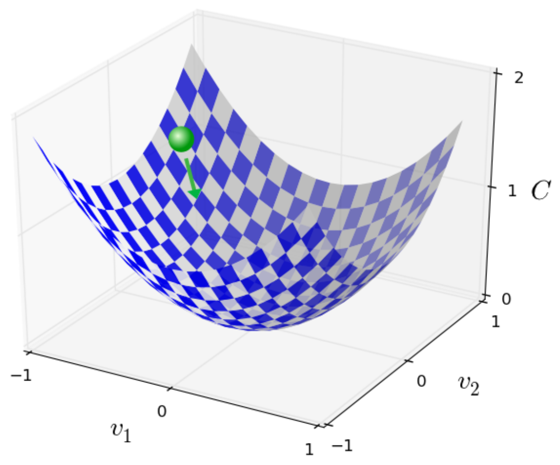


Figure 10: Neural Network Graph. [14]

To minimize the cost function, a concept called gradient descent is applied. A quadratic function has the inherit nature of being convex. Visually, it is easy to identify in Figure 10 where the cost function would be at its highest and where it would be the lowest. One method to find the minimum at the bottom of the valley is to try several values and just pick the lowest value found. However this can take a lot of computations when more dimensions are added. Instead, calculus can be applied to determine the rate of change function. For this scenario, the partial derivative of the weight with respect to the cost is calculated. Then it can be seen how the cost changes with respect to the current weight. Now when a training example is fed into the network, it can be fed into the partial derivative to determine the effect this weight has on the overall cost function. If the partial derivative is positive, then the cost function is increasing. If the partial derivative is negative, then the cost function is going downhill. By continuously training the neural network with more sample values, it can be seen which values cause the cost function to move downhill. Eventually this will converge towards an optimum value to associate with the current weight. This same concept is applied to the other weights to find the best weights for the overall function with the lowest cost.

4.1.1 Convolution Neural Network

One flavor of neural networks is Convolution Neural Network (CNN). CNN are ideal in classifying objects in images because they mimic the visual cortex found in animals. One of the unique abilities of CNNs is their inherent ability to determine their own features by looking at training images. The algorithm accomplishes this in four different stages:

1. Convolution
2. Pooling
3. Rectified Linear Units (ReLU)
4. Repeat above steps as necessary
5. Classification with a fully connected layer

The first stage is convolution. Convolution is the idea of taking several smaller subsets of the image to form features. Images can be imagined as a two dimensional array of pixel values. What the convolution step does is take a smaller two dimensional array and slides it a predetermined number of pixels to compute an array. As seen in Figure 11, a 28x28 pixel image is used. The convolution step takes a 5x5 array and uses that across the image. The stride determines number of pixels the window moves every step so not necessarily every single pixel has a convolution window applied to it. Assuming a stride step of one, the original image is condensed into a 24x24 pixel image. The idea is the condensed array will preserve the most significant parts of original image but have reduced dimensions. This forms what is called a feature map and are self learned by the neural network. Each feature map aims to determine a different part of the image that is deemed important by the algorithm. The number of feature maps that are generated is determined by the designer of the neural network.

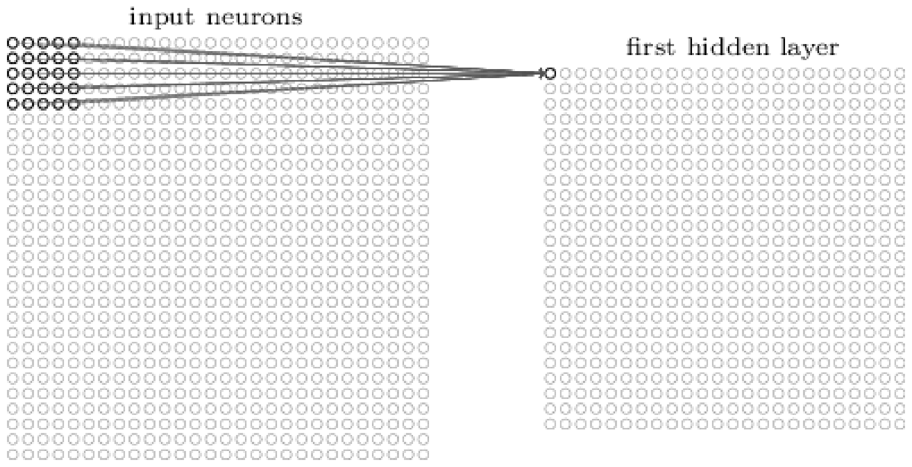


Figure 11: Convolution Step. [14]

The next stage is ReLU. ReLU is an activation function defined by f(x) = max(0, x). Basically this function removes all the negative values in the condensed image. This is done to help speed up the training process.

The next stage is pooling. Pooling aims to further reduce the complexity of the feature maps by further reducing the dimensionality. One common method used is a max pooling. Max pooling takes a small window of the feature map and takes the largest value in that window. This can be seen in Figure 12. The original image size is a 4x4 array where the pooling filter is a 2x2 array with a step size of two.

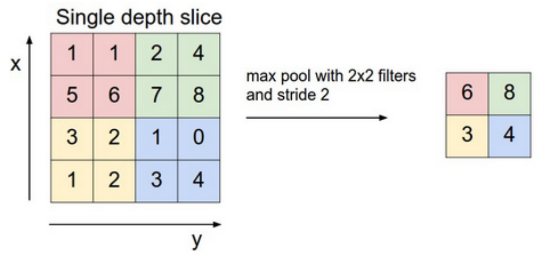


Figure 12: Max Pooling Example. [15]

The last stage is the fully connected layer. The purpose of this layer is to classify the objects based on the input. This layer takes in the inputs from the pooling layers and assigns probabilities for which object the network believes it to be.

5 Related Works

In face detection and recognition, there have been numerous proposals on how to identify faces and how to associate those faces with the correct name. Some proposals cover only face detection while others will primarily talk about associating the face with a person. Each proposal has their similarities and differences, but all of them must first identify the face through some method. Following this, they then proceed to compare this face against a set of known faces.

5.1 Viola-Jones Face Detection Algorithm

One of the earliest object classification algorithms developed was done by Paul Viola and Michael Jones and named the Viola-Jones object detection framework [6]. Originally designed for quickly identifying faces in pictures, the algorithm can be expanded to be used to detect any type of object. Their framework aimed to use machine learning to quickly detect faces with high accuracy and quickly. Their final product is an algorithm that is lightweight enough to run on a range of small and low power devices like hand held devices or embedded processors.

As explained in previous sections, their framework consisted of the Haar basis function, integral image, AdaBoost, and cascading classifiers. Their resulting system was trained with 4,916 faces and 10,000 non-faces aligned to a base resolution of 24x24 pixels. These images were found and downloaded using a random crawl of the world wide web. The detector is a 32 layer cascade of classifiers with a total of 4,297 features. Using a single 466 MHz AlphaStation XP900, the training for the 32 classifiers took weeks to finish.

Their system is able to achieve high throughput in detecting images because their first classifier consists of only two features and rejects about 60% of non-faces and detects close to 100% of faces. The second classifier contains only five features and rejects 80% of non-faces and detects almost 100% of faces. Using a 700 MHz Pentium III processor, the face detector can process a 384x288 pixels image in about 0.067 seconds.

5.2 HOG Feature Human Detection System

HOG is an object classification algorithm that is used in computer vision and image processing. It can used very effectively in detecting faces. The paper by Matt Davis and Ferat Sahin aims to combine RGB, depth, and thermal image data to detect partially obscured human faces [16]. Their paper describes using this technology with robots equipped with multiple sensors to gain real time information on the environment. Robots are becoming ubiquitous in human life and are ideal for completing tasks that humans cannot or do not want to do.

At the crux of their research is the HOG algorithm which covers color detection. This algorithm slides a detection window across a grid of cells and the gradients of the pixels in each cell are used to create a histogram of edge orientations. Objects can then be identified as a distribution of local intensity gradients. The end result is a feature vector which consists of all the feature descriptors in the image. A balance must be made between the desired detail and the size of the feature vectors. For depth, a method called Histograms of Oriented Depths (HOD) is used. HOD is exactly the same as HOG but instead of color, the depth values from the depth portion of the image are used instead of color values. Similarly, temperature data from thermal imagers can use the idea behind HOG to determine the boundary between objects with the desired temperature.

Their final set up used an Asus Xtion Prop Live camera to record RGB and depth images at a rate of 30 frames per second at a resolution of 640x480 pixels. Thermal images were recorded using an Omega OSXL-101 thermal imager at a rate of 1 frame per second at a resolution of 445x476 pixels. With this set up, a database of thousands of images with humans visible and non-visible was created to facilitate training classifiers. With these images, positive and sample sets were created and labeled to be used for training and validation. HOG features were extracted from each image and tested against a training set, validation set, and robust testing set. To further optimize extracted features, Principal Component Analysis (PCA) was used to further reduce the feature size but still maintain 99.9% of the principal components that explain the variation in the data. Four different machine learning algorithms were employed to train the individual classifiers using the feature descriptors.

1. Binary Support Vector Machine (SVM) with both linear and Radial Basis Function (RBF) kernels
2. k-Nearest Neighbors Algorithm (k-NN)
3. Naïve Bayes with a Gaussian kernel
4. AdaBoostM1 and a decision tree week learner

Each machine learning algorithm was trained with three different image types for a total of twelve classifiers. After optimizing each individual classifier, the best performing classifiers was chosen to produce a multi-layer classifier with an accuracy rate of 92.35% with a standard deviation of 1.61%. The best multi-layer classifier decision tree can be seen in Figure 13. The algorithm starts at the upper node and determines if a human is present or not in the image type. If there is a human, the algorithm moves to the right and down one level. If there is not a human, the algorithm moves to the left and down one level [16]. By using the multi-layer classifier, the accuracy was greater than eleven out of the twelve individual classifier as it did not perform better than the SVM thermal HOG classifier. However, the multi-layer classifier had a lower standard deviation than the SVM thermal HOG classifier giving it more reliability. Given a different test environment, the individual thermal classifier could potentially perform much worse than the multi-layer classifier. Given the thermal requirements, an environment with different temperatures could negatively impact the accuracy. By using depth and color as additional classifiers, the multi-layer classifier would be more tolerant to these changes.

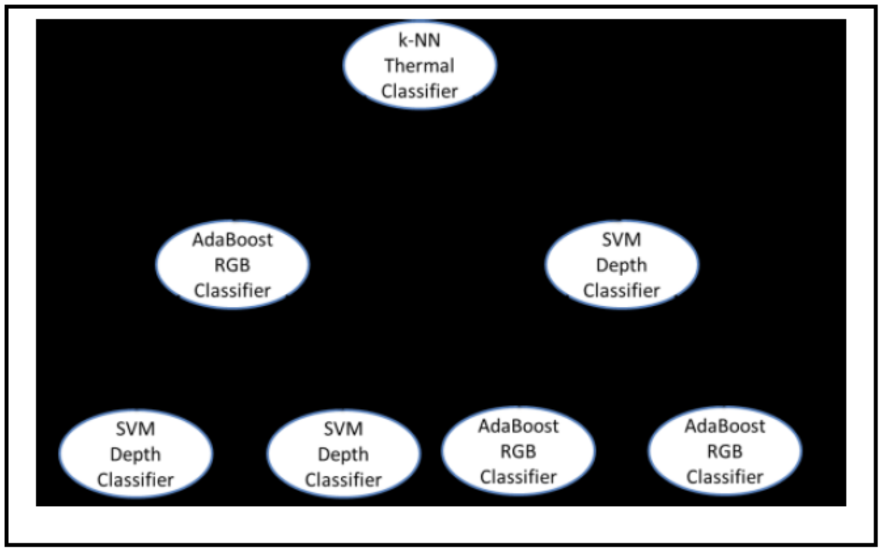


Figure 13: HOG Feature Detection Best Multi-layer Classifier. [16]

5.3 A New Face Detection Algorithm Based on Skin Color Segmentation

A skin segmentation strategy proposed by authors Yuanbo Yang, Chenggang Xie, Liebo Du, and Qin Lu aims to use skin segmentation to detect faces [17]. The authors use a face verification algorithm based on the geometric features of faces after morphological operations to widen off non-face areas. Instead of using a RGB image, the image is converted to a YCgCr color space since the authors found clustering is better in the YCgCr color space compared to the conventional YCbCr color space.

To compensate for illumination variation, the authors chose to use a strategy called gray world theory. Gray world theory assumes the average gray values of real white pixels, pixels with the top 5% of luminance values, is linearly scaled to 255, or white. Once compensated, the image can be converted to the YCgCr color space where the Y component, or luminance is ignored and the Cg and Cr colors can be used to get good clustering performance.

To detect skin in the images, the Gaussian skin color model was used. After applying the skin threshold, the authors needed to determine if faces were in a picture. When applying a threshold, other body parts like the arms or neck will show up. Therefore it is necessary to devise a strategy to differentiate skin regions to determine if the region is a face or another body part like hands. To accomplish this, noise is from the skin threshold is first removed through morphological operations. Small holes in the binary picture are then filled. The face verification algorithm then used three criteria:

1. Remove the small areas.
2. Aspect ratio of circumscribing rectangle is 0.5 to 1.8.
3. The circumscribing rectangle of the skin region is divided into three regions. The difference between the left and right parts of each section is calculated to get a ratio of the total area of white pixels. The sum of the ratios for each region totaled must be greater or equal to 0.2 to be cited as a face.

Using a test set from a laboratory and the Annotated Faces in the Wild (AFW) data set, twenty pictures were randomly selected to train the algorithm. The accuracy is 80% with an error rate of 15%. The hardware used was an Intel Core i5-4210 @ 1.7GHz 2.4GHz machine run on Windows 7 with Visual Studio 2010.

5.4 Fast Face Detection Based on Skin Segmentation and Facial Features

A skin segmentation strategy based on the YCbCr color space to determine faces in images is proposed by authors Shalini Yadav and Neeta Nain [18]. Their algorithm follows the steps:

1. Convert image from RGB to YCbCr.
2. Lighting compensation.
3. Skin color segmentation.
4. Noise removal.
5. Eyes and mouth holes detection.
6. Correct bounding box ratio and eccentricity.
7. Result of detected faces.

Because the RGB color space is not ideal for face detection due to its illumination intensity problems, the authors chose to use the YCbCr color space. Lighting variations can greatly effect skin color segmentation. To deal with darker looking pictures, the authors increased the major details of the image by adjusting the brightness of the image so the accuracy of the algorithm would be improved. The removal of noise step further increases performance. Noise can be introduced from various sources like:

- When scanning an image from film, film grain can be a source of noise.
- Damaged film.

- The mechanism for collecting data from a digital image can introduce noise.
- Electronic transmission of image data can introduce noise.

To alleviate the noise problem, the authors chose to use the Median smoothing filter. Median filtering uses the median pixel value of the neighbors to smooth the image. Using this filtering strategy, the most extreme pixel values are ignored and the sharpness of the image is preserved.

To determine where the faces are, the authors used the Euler formula to discard regions that do not have holes. This is done because the eyes and mouth will appear as holes after the skin segmentation is performed. Areas like the arms, hands, or neck will not have holes due to them being uniformly solid. To further increase the accuracy of face detection, a bounding box ratio and eccentricity is performed on the image. Since the human face has more height than it's width, a bounding box is drawn around the potential face area. The ratio between the width and height is performed and the face must pass a threshold of 0.35 to be considered a face. It is known the human face is generally oval shaped. So eccentricity can be used to try to estimate how round the detected skin region is. By calculating the eccentricity of the region, eccentricity is defined with values such that a circle is zero while a line is one. Therefore all skin regions whose eccentricity value is greater than 0.89 and less than 0.1 are discarded. This leaves the remaining eccentricity values to be more circular shaped.

Using a 2.4 GHz Intel Core i3 machined with 3GB of RAM, the authors used two different image databases for testing. The Bao database and Muct database provided single face images of a person with different orientations. Their proposed algorithm achieved an average accuracy of 96.73%.

5.5 FaceNet: A Unified Embedding for Face Recognition and Clustering

The authors Florian Schroff, Dmitry Kalenichenko, and James Philbin have proposed a system called FaceNet which uses a Deep Neural Network (DNN) to recognize faces [1]. In their paper, they present a unified face verification, recognition, and clustering system. Their method is based on a DNN which uses Euclidean embedding per image. The distances in the Euclidean embedding space can be used to determine if faces are similar with each other. Faces of the same people will tend to have similar distances as different people will have larger distances.

Face verification can then be performed by using thresholds on the distance between the Euclidean embeddings in the image. Recognition is done with a k-NN classifier. Clustering is done through k-means or agglomerative clustering. To train the DNN, the authors trained its output with a compact 128-D embedding using triplet based loss function based on Large Margin Nearest Neighbor (LMNN). The triplet is based on two matching face thumbnails and a non-matching face thumbnail. The loss separates the positive pair from the negative by the distance margin.

Two different types of DNN are used in their paper. The first model a slightly tweaked Zeiler&Fergus model which consists of multiple interleaved layers of convolutions, non-linear activations, local response normalizations, and max pooling layers. The second method is the Inception model based on Szegedy et al. which uses mixed layers that run several different convolutional and pooling layers in parallel and concatenate their responses.

The CNN is trained using stochastic gradient descent with standard back propagation. Each model is trained on a Central Processing Unit (CPU) cluster for 1,000 to 2,000 hours. The most significant gains in decreasing the error rate slows down after 500 hours, but more training can still slightly decrease the error rate. The authors explore the trade offs between the two models and found the best models are generally application dependent. The difference between the models depends on the number of parameters and Floating Point Operations Per Second (FLOPS). For a data center example, the number of FLOPS and parameters can be great due to the amount of processing power available. However for smaller devices like mobile phones, the processing power is limited and a smaller amount of FLOPS and parameters must be considered.

A total of four different image test sets were used to verify or train the different models.

1. Hold-out test set: Contained around one million image and contains disjoint identities.

2. Personal photos: A collection of manually verified photos from three personal photo collections totaling around 12,000 images.

3. Labeled Faces in the Wild (LFW): Popular academic test set. Contains around 13,000 images of faces collected from the web. 1,680 people have two or more distinct photos. [19]

4. Youtube Faces DB: Contains 3,425 videos of 1,595 different people. Used for studying the problem of unconstrained face recognition in videos. [20].

Using the hold-out test set, Table 3 shows the accuracy performance of different resolutions with the different models. Both models benefit from higher resolution images, but the validation rate is seen to be still fairly acceptable even with lower image resolutions.

Table 3: FaceNet Network Architectures [1]

Architecture	Validation Rate
NN1 (Zeiler&Fergus 220x220)	87.9% ± 1.9
NN2 (Inception 224x224)	89.4% ± 1.6
NN3 (Inception 160x160)	88.3% ± 1.7
NN4 (Inception 96x96)	82.0% ± 2.3
NNS1 (mini Inception 165x165)	82.4% ± 2.4
NNS2 (tiny Inception 140x116)	51.9% ± 2.9

Various other tests like trade offs between accuracy and number of model parameters, sensitivity to image quality, and number of images used to train the CNN was performed. The authors found that even with fewer model parameters, the Inception based model was able to achieve a comparable performance to the Zeiler&Fergus model. However both models still had a comparable number of FLOPS that were necessary. Further reduction of the model parameters is expected to further decrease accuracy and reduce the number of FLOPS necessary. But the end goal is to find other model architectures that may allow for less model parameters but not lose any accuracy.

The authors looked into the number of pixels in images and how they correlated to the validation rate. They were able find that even with images with size 80x80 pixels, or 6,400 pixels, they were still able to achieve a validation rate of 79.5%. However, at 40x40 pixels, or 1,600 pixels, the accuracy rate sharply falls off to a validation rate of 37.8%. The number of training images used shows with 2,600,000 images, the validation rate is still an acceptable 76.3%. Going upwards to 260,000,000 training images, the validation rate can go up to 86.2%.

6 Face Recognition Enhancement

My paper's proposed enhancement aims to improve the accuracy of face recognition by use of a cascaded facial detector followed by a neural network for classification. Based on the paper by Mohanty and Raghunadh [3], the first part is to improve face detection in an image. Contrary to the paper proposed by the authors, my project will use the more conventional YCbCr color space instead of their proposed YCgCr color space which proved to have better clustering performance. To achieve better face detection, a three layer approach is used to determine where faces are. By applying HOG, skin segmentation, and Viola-Jones object detection onto a face, a better face detection scheme is achieved.

To recognize faces and associate them with an identifier, the improved face detection algorithm can be used in conjunction with a face recognition technique. A better face detection algorithm will ensure the detected face in the image is actually a real face rather than a false positive. One way of classifying objects in images is to use neural networks. One of the best types of neural networks for classification is CNN. In summary, this paper presents an enhanced algorithm for detecting faces and uses a neural network to associate the faces with the person's actual identity.

7 Tools

To achieve the desired result, several different technologies were used in conjunction. All the tools and data used were open source and free to use by researchers and/or developers.

7.1 Open Source Computer Vision

OpenCV is an open source library of computer vision and machine learning software [21]. The library is licensed under the BSD license, making it free to use and open to modification by third parties. Due to its open license, glsopencv has more than 47,000 users in their open community and an estimated 14 million downloads. OpenCV is used in many well established entities including Google, IBM, Microsoft, government bodies, and research groups.

The library has various algorithms that includes but is not limited to helping users detect and recognize faces, perform morphological operations, convert between color spaces, and finding shapes and holes in objects. Practical implementations range anywhere from detection of swimming pool drowning accidents in Europe and checking runways for debris in Turkey.

OpenCV is primarily developed in C++, but has support in C, Python, Java, Matlab. Operating systems supported are Windows, Linux, Android, and Mac OS. OpenCV is generally used for real time vision applications and takes advantage of CPU instructions like MME and SSE when available. Due to the growing popularity of using graphic cards to perform parallel processing, a full featured CUDA and Open Computing Language (OpenCL) interfaces are being developed.

7.1.1 Morphological Operations

OpenCV provides a morphological library to help warp images as necessary. Morphological operations take a image and a structuring element as input and combines them with a set operator like intersection, union, inclusion, and complement [22]. By using just the color space threshold, imperfections arise in the image like noise or texture. Morphological operations help smooth out and reduce the noise by looking at neighboring pixels and using the structuring element to obtain the desired shape.

The structuring element is the desired shape of the final image. The structuring element can be anything like ellipses, rectangles, or plus signs. These shapes are generally represented as a grid in Cartesian coordinates as seen in Figure 14. The structuring element is placed over each pixel and its surrounding neighbors to determine if that pixel area fits the structuring element. Depending on the operation used, the pixel value at that point is changed.

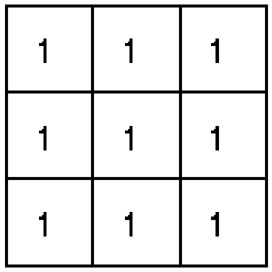


Figure 14: 3x3 Square Structuring Element. [22]

In addition to the structuring element, the four most common morphological operations are dilation, erosion, opening, and closing. Each operation aims to perform a different task on the image. Each operation uses the structuring element as the basis for what the mask should be. Dilation will grow image regions. Erosion will shrink image regions. Opening will make holes in the image more prominent. Closing removes holes in the images. Each operation may be applied to an image multiple times to help obtain a better image.

7.1.2 Contours

Finding contours is another useful set of functions provided by OpenCV. As explained by the tutorials in their documentation, a contour is a curve joining all the continuous points along the boundary having the same color or intensity [21]. In other words, contours are useful for detecting unbroken shapes in images. In identifying shapes, functionality is provided to determine how much detail is needed. For example, the contour of a straight line can have every single point on the line or just the two endpoints. Other shapes like rectangles will also possess the same attribute where all the points along the square or only the vertices are stored. By storing just the minimal information required for the shape, this has great potential for saving memory and retaining just the most important points for a simple shape.

In addition, functions in OpenCV provide ways to detect shapes inside shapes, thus creating a hierarchy of the shapes. For example, as seen in Figure 15, OpenCV will identify shapes in the object but also have identifiers relating the parent and child.

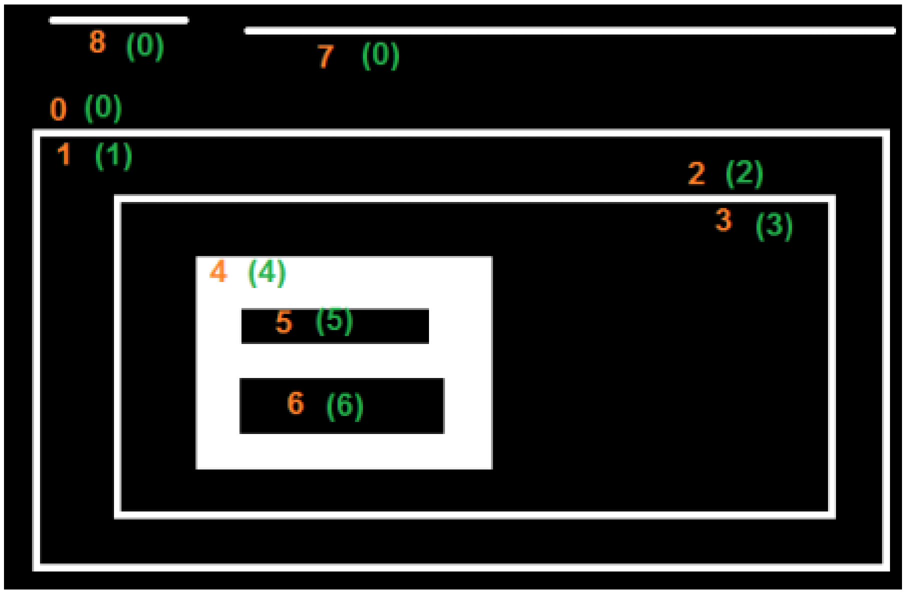


Figure 15: OpenCV Hierarchy Example. [21]

As indicated by Figure 15, all the hierarchy levels are noted in the parenthesis and the contour identifier on the left of the parenthesis. Contours 0, 7, and 8 are considered hierarchy-0 as they are the outermost shapes. Contour 0 and contour 1 may seem to reference the same image, but they refer to the inner and outer part of the contour. Therefore contour 1 is considered a child of contour 0. All the parent $\leftrightarrow$ child information is preserved and available for the developer to use as needed in a convenient hierarchy array.

7.2 dlib

dlib is a C++ toolkit containing machine learning algorithms and tools for creating complex software in C++ to solve real world problems [23]. dlib is open source and licensed via the Boost Software License and is available free of charge for anyone to use in any application. Some major features of dlib include machine learning algorithms, numerical algorithms, graphical model inference algorithms, image processing, threading, networking, graphical user interfaces, and data compression and integrity algorithms.

7.3 Torch

Torch is an open source scientific computing framework with support for machine learning algorithms [24]. Torch uses the scripting language LuaJIT and is implemented with CUDA underneath. It is a popular tool that helps users design and maintain neural nets for whatever problem they are trying to solve. This project is used in OpenFace to construct the neural network.

7.4 OpenFace

OpenFace is an open source Python and Torch implementation of face recognition with deep neural networks [2]. The source code of the Torch and Python model files are copyrighted by Carnegie Mellon University and licensed under the Apache 2.0 License unless otherwise stated. The project is based on the paper FaceNet: A Unified Embedding for Face Recognition and Clustering by Florian Schroff, Dmitry Kalenichenko, and James Philbin at Google. The research was supported by the National Science Foundation and additional support was provided by Intel Corporation, Google, Vodafone, NVIDIA, and the Conklin Kistler family fund. The OpenFace platform uses the following steps to perform face recognition and is further summarized in Figure 16:

1. Detect faces with pre-trained models from dlib or OpenCV.

2. Transform the face for the neural network.

3. Use a DNN to represent the face with a 128-dimensional unit hypersphere.

4. Apply a clustering, classification, or similarity detection technique to the features to determine who the face belongs to.

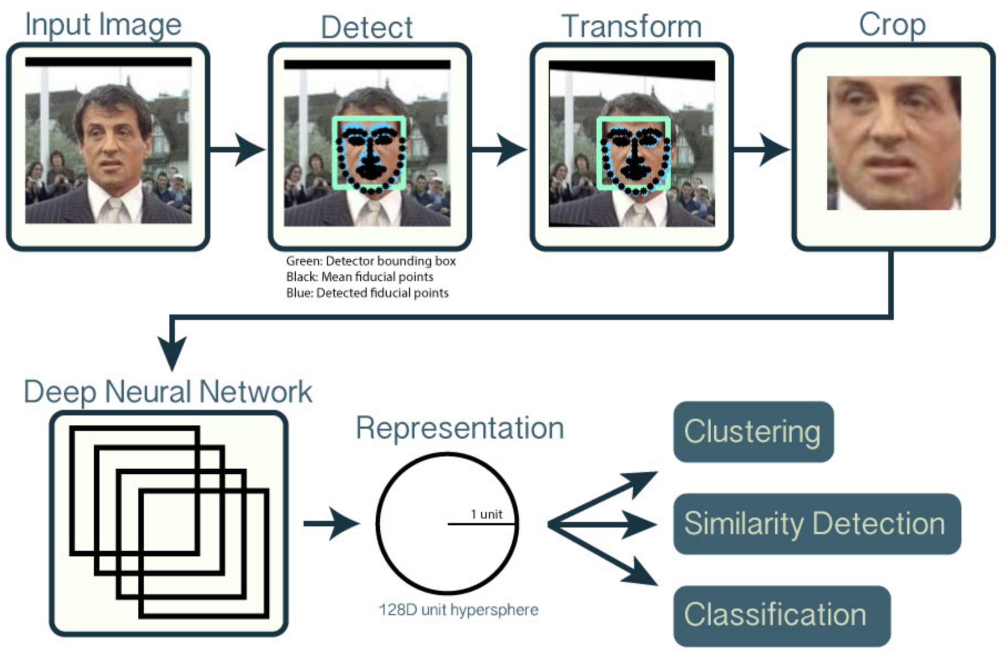


Figure 16: OpenFace Summary. [2]

To prepare an image for facial recognition, OpenFace transforms the image to meet the requirements needed for the neural network in a process called alignment. Alignment is the process in which the face is located, transformed, and then resized into a uniform shape. In the real world, faces in images may have different poses or angles in which they are positioned. This makes taking measurements for faces slightly more difficult. If a face was trained solely with frontal facing images and an image of the same person but different pose was given during identification, the neural net might generate inaccurate measurement values for this particular pose. This can lead to a different prediction by the classifier which results in a false negative or true negative depending on the classifier's confidence threshold. To help deal with this problem, OpenFace transforms the face via OpenCV affine transformations and dlibs real time pose estimation which warps the face and attempts to make it a frontal facing image. OpenFace then tries to align the face such that 68 measurements called landmarks on the face can be seen. The landmarks together produce a generic structure representing the human face so the eyes, nose, and mouth appear in a general location as seen in Figure 17.

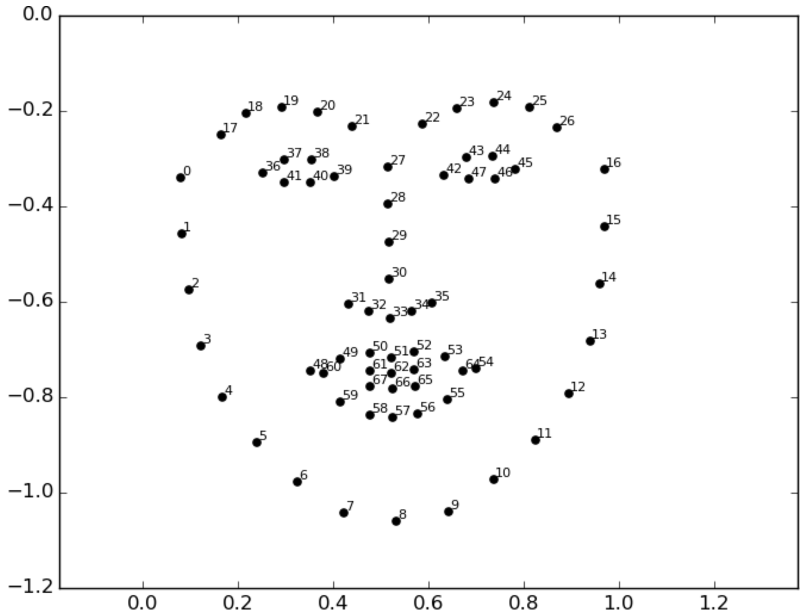


Figure 17: OpenFace Landmarks. [2]

OpenFace provides several pre-trained neural network models based on Google's FaceNet trained with the publicly available FaceScrub and CASIA-WebFace face databases. Each model requires the faces to be the best frontal faces possible which is achieved by the alignment method mentioned earlier. The number of parameters and accuracies can be seen in Table 4. The different model's parameters affects the time it takes to train the neural network and how many layers it has. More parameters will increase the time. Generally more parameters will increase the accuracy of the neural network, but OpenFace's models shows that cutting down on nearly half the parameters can still boast an impressive accuracy rate. All the models in Table 4 are trained against the LFW data set. The networks are designed according to Google's FaceNet NN4 networks with the networks with "small" appended to them having certain layers removed.

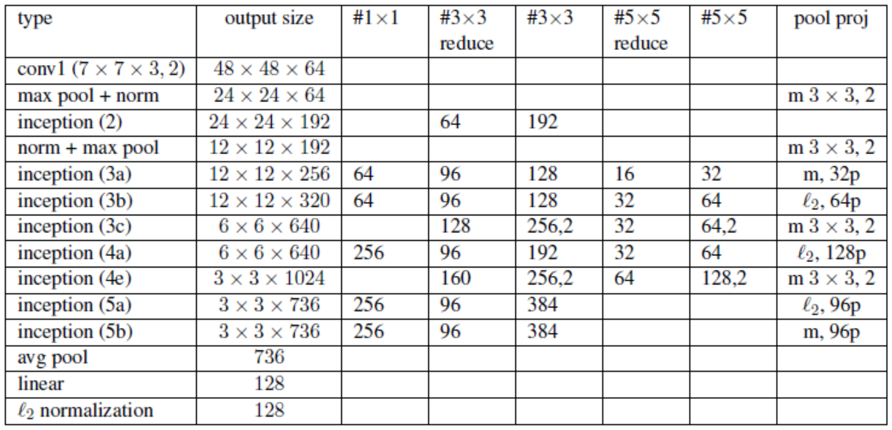
Table 4: OpenFace Pre-trained Models [2]

Model	Number of Parameters	Accuracy (%)
nn4.v2	6959088	91.57 ± 1.52
n4.small1.v1	5579520	92.10 ± 1.60
nn4.small2.v1	3733968	92.92 ± 1.34

After going through layers in the neural network, OpenFace's final result is a

128-dimensional generic representation of anyone's face. An unique quality of this representation is larger distances between two face embeddings indicates the two faces are likely not the same people. This allows clustering, similarity detection, and classification tasks to be easier since the Euclidean distance between features is more meaningful.

Figure 18 shows the modified nn4 network designed for the smaller data size used to train the neural network. According to the training options, the network was run with 1000 epochs with 250 batches per epoch. In each batch, 20 images are taken from 15 people. This generates the 128 embeddings. The triplet losses are generated from these embeddings. These triplet losses are forward passed through the network. The triplet losses are then backpropagated to further optimize the output. If the triplet losses does not pass a threshold, it is discarded.



type	output size	#1×1	#3×3 reduce	#3×3	#5×5 reduce	#5×5	pool proj
conv1 ($7 \times 7 \times 3, 2$)	$48 \times 48 \times 64$						
max pool + norm	$24 \times 24 \times 64$						m $3 \times 3, 2$
inception (2)	$24 \times 24 \times 192$		64	192			
norm + max pool	$12 \times 12 \times 192$						m $3 \times 3, 2$
inception (3a)	$12 \times 12 \times 256$	64	96	128	16	32	m, 32p
inception (3b)	$12 \times 12 \times 320$	64	96	128	32	64	ℓ_2, 64p
inception (3c)	$6 \times 6 \times 640$		128	256,2	32	64,2	m $3 \times 3, 2$
inception (4a)	$6 \times 6 \times 640$	256	96	192	32	64	ℓ_2, 128p
inception (4e)	$3 \times 3 \times 1024$		160	256,2	64	128,2	m $3 \times 3, 2$
inception (5a)	$3 \times 3 \times 736$	256	96	384			ℓ_2, 96p
inception (5b)	$3 \times 3 \times 736$	256	96	384			m, 96p
avg pool	736						
linear	128						
ℓ_2 normalization	128						

Figure 18: OpenFace nn4.small2 Network Definition. [2]

7.5 FEI Face Database

The FEI face database is a Brazillian database of images taken between June 2005 and March 2006 at the Artificial Intelligence Laboratory of FEI in São Bernardo do Campo, São Paulo, Brazil [25]. There are 200 unique people with 14 images per person. All images are size 640x480 pixels. All images are in color and taken against a white background with a full frontal face accompanied with profile rotations of up to 180 degrees. In addition, there are pictures with different facial expressions and one picture with illumination changes. Each face is represented by students and staff at FEI with ages between 19 and 40 years old. Each person has unique appearances, hairstyles, and adornments. The number of female and male subjects are 100 each. An example of one set of images can be seen in Figure 19. Each of the images are labeled with an identifier. The top left is labeled as image 1. Going left to right on the top row, the last image is labeled image 7. The bottom left is labeled as image 8. Going left to right on the bottom row, the last image is labeled image 14.

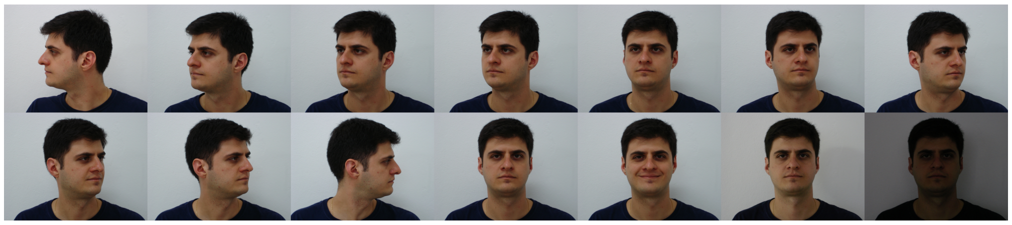


Figure 19: FEI Face Example. [25]

8 Implementation

My project was done in an lubuntu 16.10 Yakkety virtual machine environment. The hardware used for the virtual machine are two virtualized CPU cores with 2048GB of RAM. The underlying system is a Windows 10 environment with an Intel Core i5-6600K Quad Core CPU @ 3.50 GHz, 16GB of RAM, and AMD Radeon R9 380 video card. The hardware was chosen because it was available for use with this project.

The implementation of this project can be summarized into four parts and can be visually seen in Figure 20.

1. Obtain the image.
2. Feed the image into a cascaded face detector.
3. Use the detected face and feed that into the neural network and classifier.
4. Person is identified.

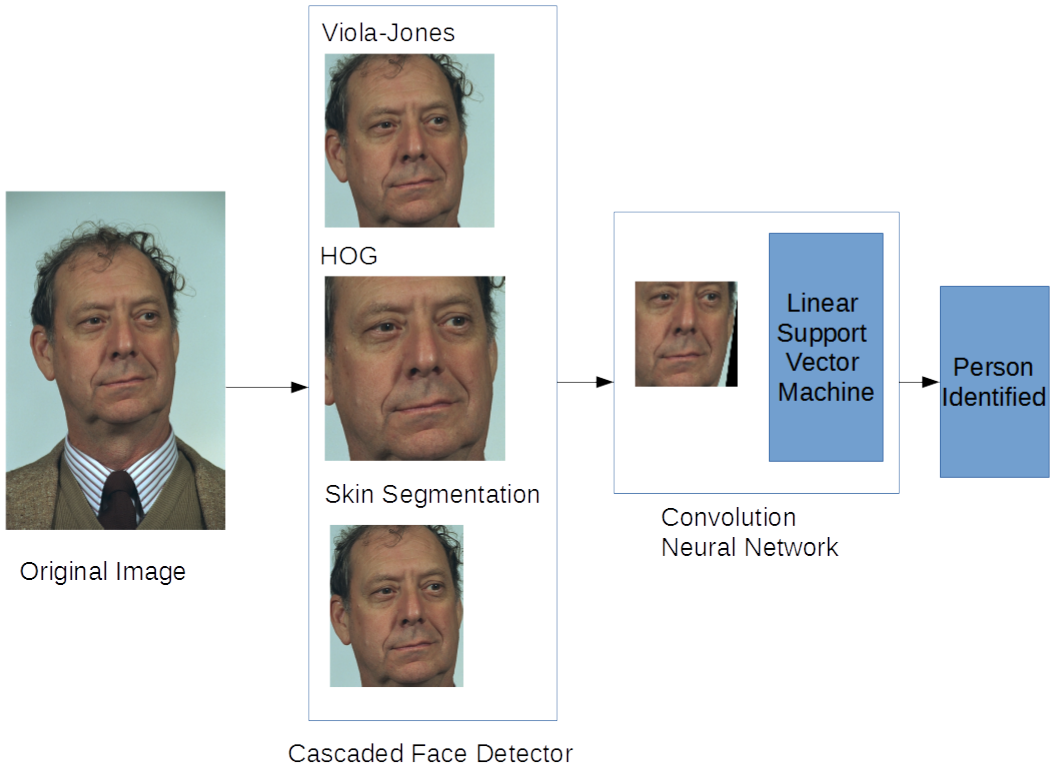
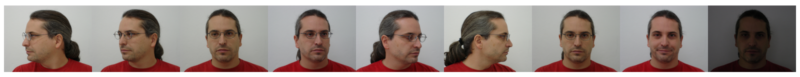
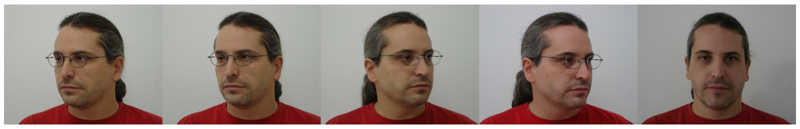


Figure 20: Summary of Project. [26][27]

The first step is to obtain an image from a face database. Due to the requirements of skin segmentation, only color photos can be used as skin tone color is important in determining where the face is. Non-color photos have to be filtered from the database before they can be used effectively. In addition, the face database should have at least two or more (more is better) faces of the same person. This is necessary because the classifier will need to be trained with images of the person. A second validation set of images will then be used to determine whether the system can correctly identify the person. This project uses the FEI Face Database which is fully colored and contains fourteen images of the same person. The first nine images consists of the person's frontal face and profile This project took 9 images for training and 5 images for validation. For each image set, the same face orientation and expression images were chosen. The images were divided so that at least one profile, frontal, and expression image was chosen to be in the training set. An example set is shown in Figure 21. Images labeled 3, 4, 7, 8, and 13 were used for validation. Images labeled 1, 2, 5, 6, 9, 10, 11, 12, and 14 were used for training.



(a) Training Set



(b) Validation Set

Figure 21: FEI Faces Split Between Training and Validation Set

The second step involves implementing the HOG, Viola-Jones, and skin segmentation algorithm. This is done with the aid of OpenCV and dlib which provides many computer vision functions. Each algorithm is designed for detecting multiple faces in an image. For the purposes of this project, only one person will ever be present in the images used. Multiple faces detected in my case would indicate either an incorrectly detected object as a face or a duplicate of a correctly detected face. To solve this issue, the image with the greatest amount of area is selected as the image with the most accurate face. Parts of the body like the nose and mouth are occasionally picked up and incorrectly identified as faces with Viola-Jones. Since the area of a person's full face has to be larger than these features, it was reasoned the biggest object detected then would most likely be the face. This would not hold true if there are multiple objects in the background or other major body parts in the photo. The database used however does not have these extraneous objects.

For both Viola-Jones and skin segmentation, it is necessary to convert the RGB images to different color spaces. For Viola-Jones, the image needed to be in grayscale. OpenCV provides functionality for this and the resulting converted image can be seen in Figure 22. HOG is color agnostic and does not need a color space conversion.

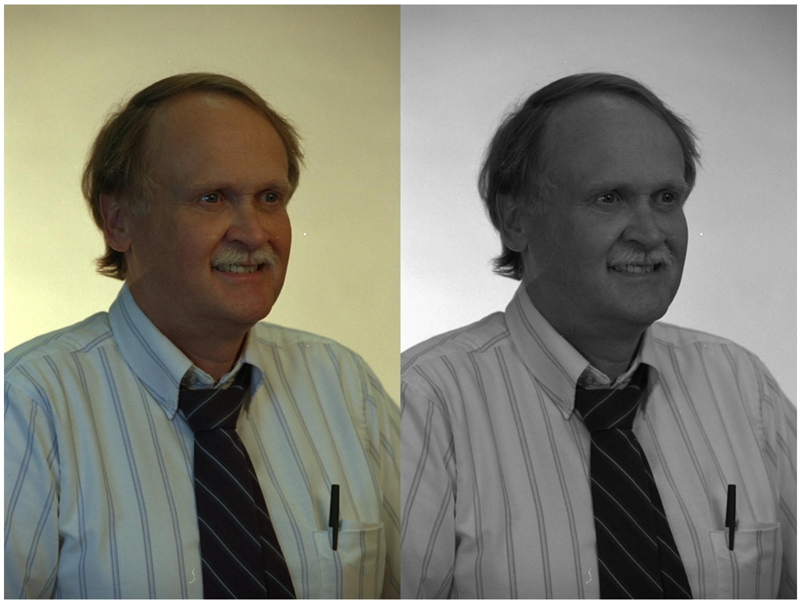


Figure 22: OpenCV RGB Image to Grayscale. Left image is the original image. Right image is the gray scale image. [26][27]

For skin segmentation, the color space used is YCrCb. OpenCV provides conversion functions for this as well and the resulting image split into the Y-component, Cb-component, and Cr-component can be seen in Figure 23.

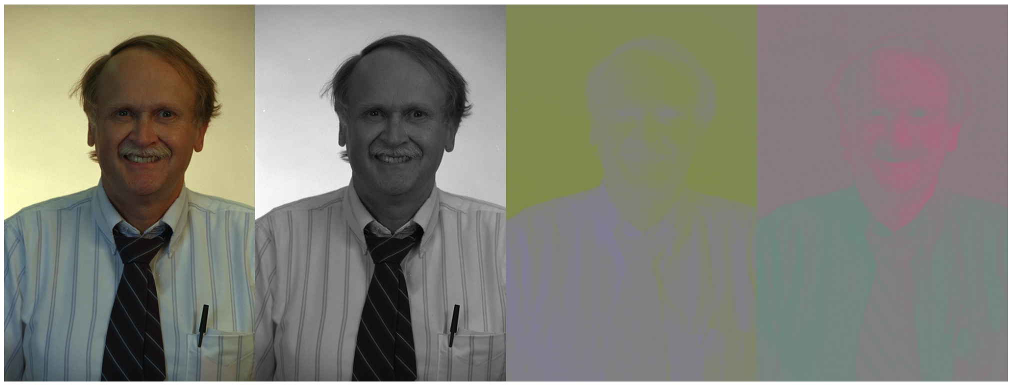


Figure 23: OpenCV RGB Image to YCbCr. The images are from left to right: Original Image, Y-component Image, Cb-component Image, and Cr-component Image. [26][27]

For HOG, dlib provides a pre-trained HOG frontal face detector. The HOG detector is trained with about 3000 images from LFW. The dlib library will make use of SSE CPU instructions if available. The final trained HOG detector can be seen in Figure 24. As seen, it resembles the front face of a human being.

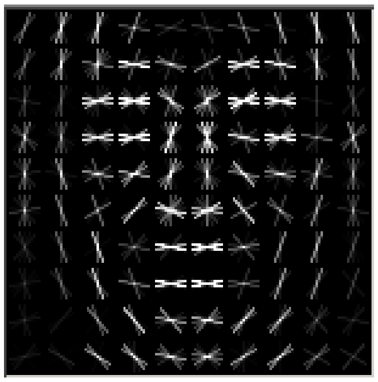


Figure 24: dlib Frontal Face HOG Detector [23]

For future unknown images, dlib proceeds to compute the gradient vector representing the image. This gradient vector is then fed into a linear SVM to determine how close it is to the trained face. If the threshold return by the classifier exceeds dlib's minimum threshold, the face's coordinates are recorded and and returned back to the caller.

For Viola-Jones, OpenCV has prebuilt models for which Haar features are used in finding objects which include things like frontal faces, profile faces, full body, and eyes. For this project, a combination of the default frontal and profile face models are used in combination for finding faces. The default frontal face model consists of 25 cascade stages. The profile face model consists of 26 cascade stages. This is done to improve the accuracy in detecting not just frontal faces but profile faces as well. The face database used is known to contain at least two full profile images per person. Since the algorithm is designed to find one or more faces and I am using two different Haar features in conjunction, it is possible for more than one face to be detected. Duplicates are removed via the greater area method mentioned before.

One other parameter worth mentioning with OpenCV Viola-Jones implementation is the ability to tweak the sensitivity of objects detected as faces. This parameter is called the minimum neighbors and it attempts to eliminate false positives by setting thresholds. As seen in Figure 25, the center image has the minimum neighbor set to 1. It detects two areas which are clearly not faces since it identified part of the man's shirt as potential faces. By increasing the minimum neighbors parameter, it forces the algorithm to detect more face candidates according to the threshold at that location. This project's minimum neighbor threshold is set to 3. Upping the minimum neighbors requirements to 3 eliminates many of the false positives but does increase the risk of false negatives.

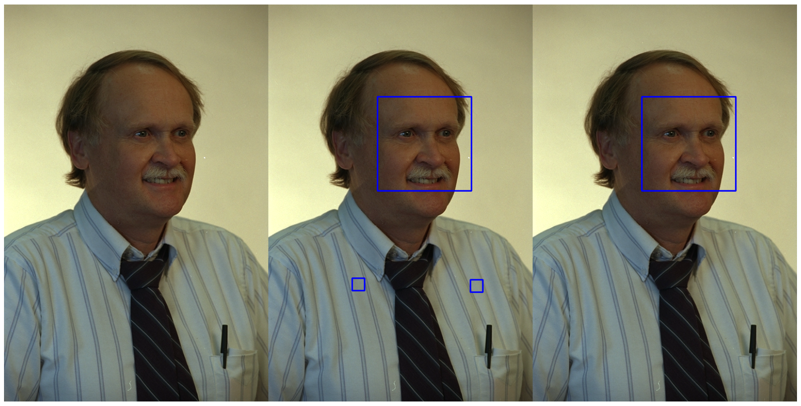


Figure 25: OpenCV Haar Cascades Minimum Neighbors: The far left image is the original image. The center image's minimum neighbor is set to 1. The far right image's minimum neighbor is set to 3. [26][27]

For skin segmentation, OpenCV does not have any specific prebuilt models. However, there are functions that support building a skin segmentation algorithm. OpenCV provides a threshold function for different color spaces. By applying the threshold function, a binary image is generated indicating which areas are there with white and areas that are not there with black. For this project, the threshold used was:

$$30 < Y < 255, 133 < Cr < 173, 77 < Cb < 127 \tag{9}$$

After applying the threshold to the image, additional operations was applied to further smooth the image. OpenCV provides access to morphological and Gaussian blur operations. The morphological operations used was erode and dilate with an elliptical structuring element with an iteration of five times. To further smooth the image and remove pixelated areas, Gaussian blurring was applied. The sequence of image modifications can be seen in Figure 26.

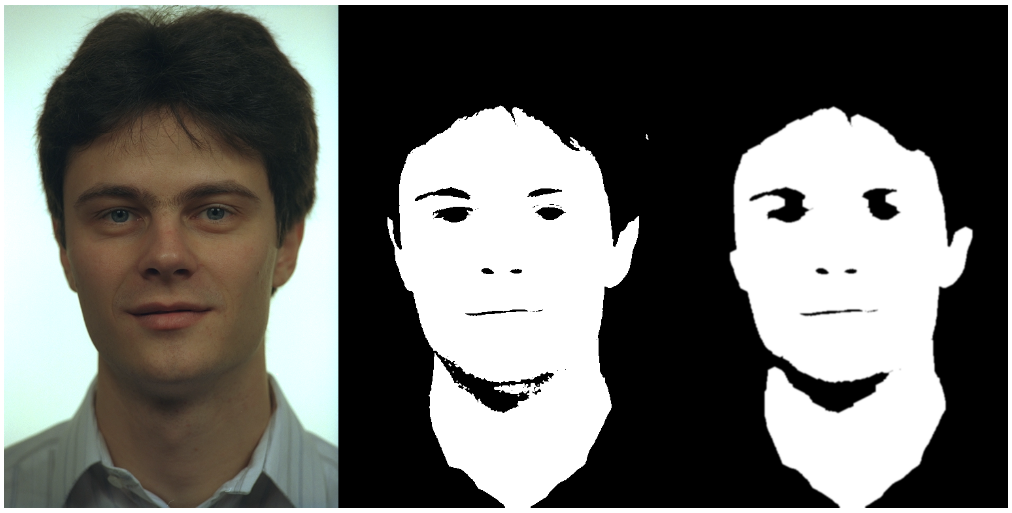


Figure 26: Skin Segmentation With Threshold And Morphological Operations: The far left image is the original image. The center image is with the threshold applied. The far right image is with morphological operations applied. [26][27]

The next step is to identify interesting shapes in the image. The shapes this project focuses on are eyes. OpenCV provides additional assistance in locating these features by finding contours in the image. To filter out contours that are extremely small and unlikely to be faces, there is a minimum area the contour must pass before being used. Once this area threshold is applied, the resulting contours can include more things than just the face. An example run is seen in Figure 27 where three contours with the minimum area are shown. Each of the contours represent the full face and each of the two eyes.

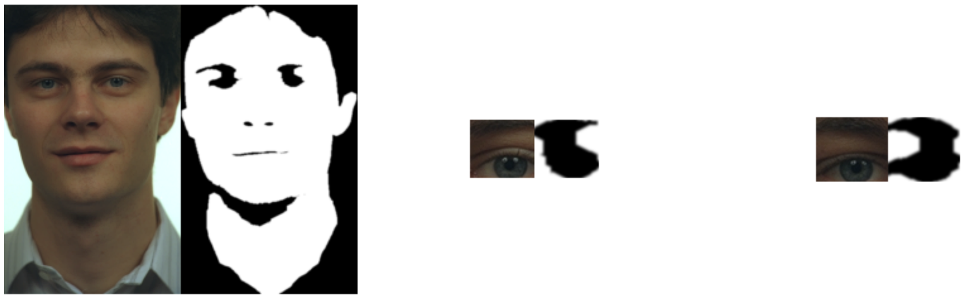


Figure 27: Skin Segmentation Contour Results: The three contours detected by OpenCV showing the face and the two eyes. The left image in each section represents the original image and the right image represents the threshold applied. [26][27]

To further narrow down where the face is, contours with small areas are removed and the contour of the face must be separated from other contours like the mouth, eyes, or nostril openings. This was achieved by looking at the hierarchy of contours. As seen, the face contour actually contains child contours that represent things like the eyes. So another threshold is applied where the number of child contours must be greater than one representing the eyes. The low threshold number is to account for profile faces where the second eye may not form an individual contour. Other detected contours like the eyes or mouth are unlikely to have a child contour. Generally, only the face will have multiple contours and not the eyes or mouth. Each contour with at least one child is then determined to be a candidate for a face. Duplicates are removed by the method mentioned above.

The third step uses a combination of OpenFace and scikit-learn [28] to produce the convolution neural network and classifier. OpenFace is a convolution neural network platform that has pre-trained models for detecting faces. The pre-trained models are trained with the FaceScrub and CASIA-WebFace face databases. The models generate 128 measurements to be used with a classifier. There are several tools available in OpenFace that helps generate a training set summary file to be used with a classifier. The model used for this project is OpenFace's default face model "nn4.small2.v1.t7" consisting of 26 layers. An example of one of the layer's features is seen in Figure 28. The layer shown is the third layer of the 26 layers and is the ReLU layer consisting of 64 features.

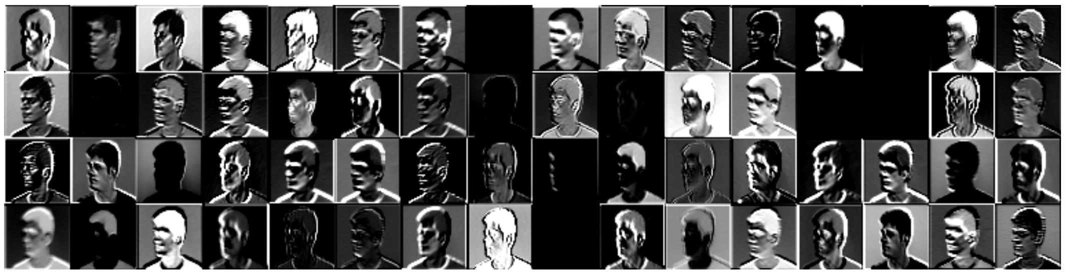


Figure 28: OpenFace Layer 3 ReLU Layer

The images are aligned to 96x96 pixels and warped to ensure the landmarks are visible. An aligned example can be seen in Figure 29 where the full sized image is on the left and the aligned image is on the right. The image is aligned to allow for a uniform embedding to be generated to be fed into the neural network's generated measurements.

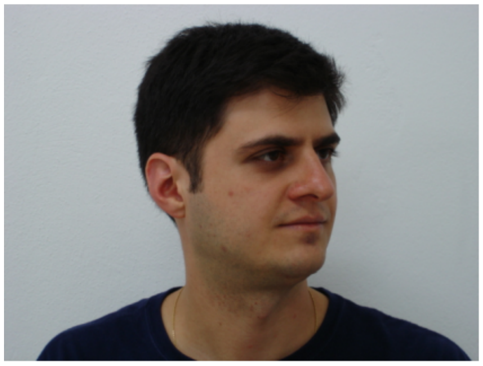
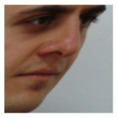



Figure 29: OpenFace Aligned Image: The right image is the original image. The left image is the aligned image.

Certain tools and scripts provided by OpenFace have been tweaked slightly to fit my project's requirements. One of the scripts provided uses HOG to perform face detection. Since my project performs the face detection, this part of the script has been adjusted to accept the faces my algorithm detects and aligns them as necessary for the neural network. Once the faces have been aligned, the next step is to feed them into the neural network and generate the embeddings. These generated embeddings represent the 128 measurements that will be fed into the classifier.

The classifier used for this project is scikit-learn's linear SVM. To begin using the classifier, the images for each labeled person in the face databases have been randomly split into a training and validation set. The training set is fed into the linear SVM and it begins to construct the best hyperplane to separate each person based on the 128 measurements. The trained linear SVM is saved in a python pickle file, which is a python module for serializing and de-serializing a python object, so the training does not need to be done again for future cases. The next step is feeding the aligned validation images from my face detection algorithm into a classifier to generate a list of probabilities. Each probability has the the person's identifier associated with it. To determine who the person's face belongs to, the highest probability is chosen which should ideally be the actual person's identity. Otherwise, this will be known as a false detection.

Seven combinations were selected to test how well each combination would detect and recognize a face. The first 4 are various forms of the the individual algorithms cascaded. The seven combinations are:

1. HOG $\rightarrow$ Viola-Jones $\rightarrow$ Skin Segmentation
2. HOG $\rightarrow$ Viola-Jones
3. HOG $\rightarrow$ Skin Segmentation
4. Viola-Jones $\rightarrow$ Skin Segmentation

5. HOG
6. Viola-Jones
7. Skin Segmentation

9 Results

Based on the implementation from section 8, the results are tabulated in the following tables. Based on the paper by Mohanty and Raghunadh [3], Detection Success Rate (DSR) and False Detection Rate (FDR) are calculated to determine the accuracy of the face detector. Error Rate (ER) is included and is the complement of the DSR.

DSR is defined as the number of detected faces over the total number of faces. The number of detected faces is determined by how many faces the algorithm detected, whether it be a true positive or a false positive. Higher is better.

$$DSR = (Number\ of\ Correct\ Detection/Total\ Number\ of\ Faces) * 100 \quad (10)$$

ER is defined as the number of faces that were not identified but were present in the image over the total number of faces. Lower is better.

$$ER = (Number\ of\ Faces\ Not\ Detected/Total\ Number\ of\ Faces) * 100 \quad (11)$$

FDR is defined as the number of falsely detected faces over the total number of detection. The number of falsely detected faces is determined by dlib's ability to align the face. If dlib is able to find landmark indices after the face has been warped, then the face candidate is considered a face. If dlib is unable to find landmark indices, then the face is flagged as a false alarm. This can also happen if say the pocket in an image is incorrectly identified as a face. Lower is better.

$$FDR = (Number\ of\ False\ Detections/Total\ Number\ of\ Faces) * 100 \quad (12)$$

Table 5 and Table 6 shows the accuracy comparisons between the three databases.

Table 5: Face Detection Accuracy with FEI Face Database

Detection Method	Total Faces	Hits	Misses	False Alarms
HOG $\rightarrow$ Viola-Jones $\rightarrow$ Skin Segmentation	1000	929	71	3
HOG $\rightarrow$ Viola-Jones	1000	969	31	0
HOG $\rightarrow$ Skin Segmentation	1000	954	46	3
Viola-Jones $\rightarrow$ Skin Segmentation	1000	934	66	3
HOG	1000	1000	0	1
Viola-Jones	1000	974	26	9
Skin Segmentation	1000	956	44	4

Table 6: Face Detection Accuracy Percentages with FEI Face Database

Detection Method	DSR (%)	FDR (%)	ER (%)
HOG $\rightarrow$ Viola-Jones $\rightarrow$ Skin Segmentation	92.9	0.3	7.1
HOG $\rightarrow$ Viola-Jones	96.9	0	3.1
HOG $\rightarrow$ Skin Segmentation	95.4	0.3	4.6
Viola-Jones $\rightarrow$ Skin Segmentation	93.4	0.3	6.6
HOG	100	0.1	0
Viola-Jones	97.4	0.9	2.6
Skin Segmentation	95.6	0.4	4.4

The average and median time it took to execute the proposed algorithm is included in Table 7.

Table 7: Face Detection Performance with FEI Face Database

Detection Method	Average Execution Time (ms)	Median Execution Time (ms)
HOG $\rightarrow$ Viola-Jones $\rightarrow$ Skin Segmentation	44.9	44.8
HOG $\rightarrow$ Viola-Jones	44.2	44.2
HOG $\rightarrow$ Skin Segmentation	31.7	31.6
Viola-Jones $\rightarrow$ Skin Segmentation	26.3	26.2
HOG	31.8	31.6
Viola-Jones	26.2	26.0
Skin Segmentation	0.5	0.4

Similar to face detection, DSR and ER will be used for the face recognition accuracy. However, the definition will be slightly tweaked to test against the total number of aligned faces versus total number of faces. The linear SVM will always output its best guess as to who the person is. Since no threshold for the confidence score is used, FDR is not relevant and has been excluded. The results for face recognition can be seen in Table 8 and Table 9.

$$DSR = (Number\ of\ Correct\ Detection/Total\ Number\ of\ Aligned\ Faces) * 100 \tag{13}$$

$$ER = (Number\ of\ Incorrect\ Detections/Total\ Number\ of\ Aligned\ Faces) * 100 \tag{14}$$

Table 8: Face Recognition Accuracy With FEI Database

Detection Method	Total Faces	Total Faces Aligned	Hit	Misses
HOG → Viola-Jones → Skin Segmentation	1000	926	755	171
HOG → Viola-Jones	1000	969	786	183
HOG → Skin Segmentation	1000	951	768	183
Viola-Jones → Skin Segmentation	1000	931	762	169
HOG	1000	999	803	196
Viola-Jones	1000	965	788	177
Skin Segmentation	1000	952	779	173

Table 9: Face Recognition Accuracy Percentages with FEI Face Database

Detection Method	DSR (%)	ER (%)
HOG → Viola-Jones → Skin Segmentation	81.5	18.5
HOG → Viola-Jones	81.1	18.9
HOG → Skin Segmentation	80.8	19.2
Viola-Jones → Skin Segmentation	81.8	18.2
HOG	80.4	19.6
Viola-Jones	81.7	18.3
Skin Segmentation	81.8	18.2

Associated with face recognition are the linear SVM results which have a confidence score. The confidence score dictates how much the system believes the face belongs to that person's identity. This confidence score is influenced by the number of people trained and the number of feature vectors given. The SVM gives each identity a score but should have low scores for negative identities and a high score for the positive identity. The sum of all the confidence scores totals to 100%. Below is the top three average confidence scores. Higher is better for the 1st confidence score. Lower is better for the 2nd and 3rd confidence scores.

Table 10: Face Recognition Accuracy Average Confidence Scores with FEI Face Database

Detection Method	1st Highest Confidence (%)	2nd Highest Confidence (%)	3rd Highest Confidence (%)
HOG → Viola-Jones → Skin Segmentation	5.3	2.4	1.5
HOG → Viola-Jones	5.3	2.5	1.6
HOG → Skin Segmentation	5.3	2.5	1.5
Viola-Jones → Skin Segmentation	5.3	2.4	1.6
HOG	5.3	2.5	1.6
Viola-Jones	5.3	2.5	1.6
Skin Segmentation	5.3	2.5	1.6

Each algorithm combination decides whether or not it detected a face in the image. How often each of the seven combinations agreed a face was in the picture is shown in Figure 30. How often each of the three individual algorithms agreed with each other is shown in Figure 31. Higher is better.

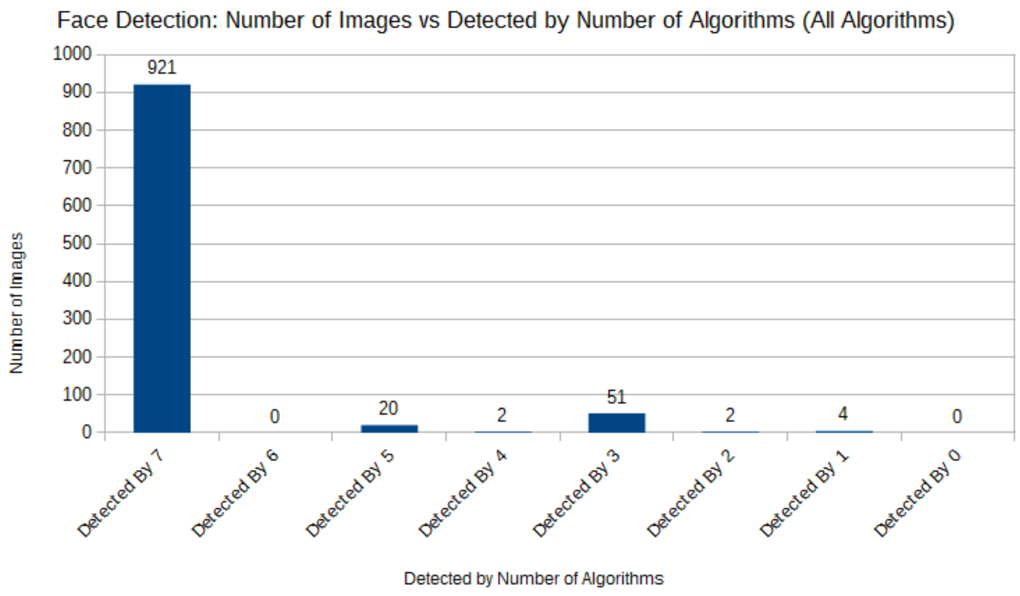


Figure 30: Face Detection: Number of Images vs Detected by Number of Algorithms (All Algorithms)

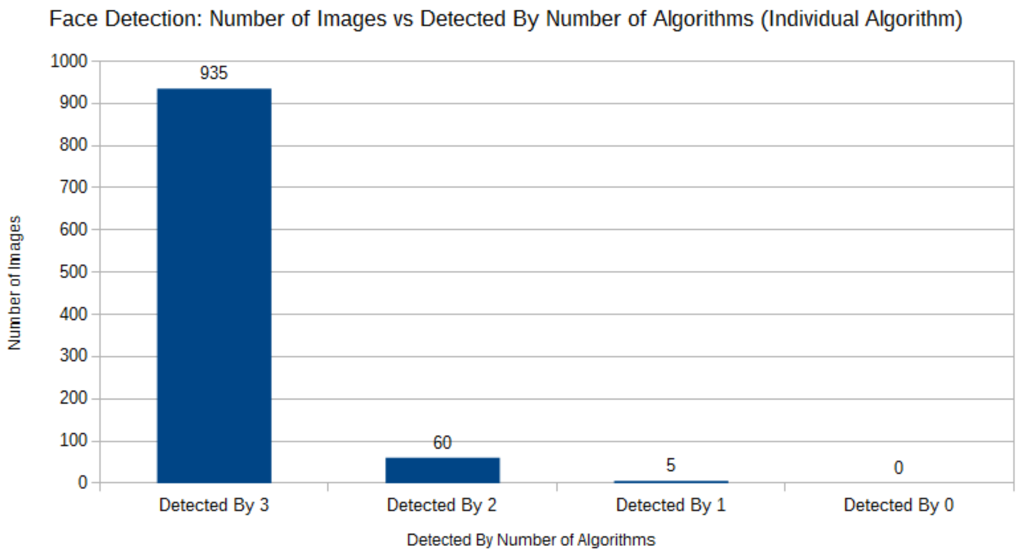


Figure 31: Face Detection: Number of Images vs Detected by Number of Algorithms (Individual Algorithms)

Figure 32 shows how often each of the seven combinations correctly recognized the person in an image. Figure 33 shows how often each of the individual algorithms correctly recognized the person in an image. Higher is better.

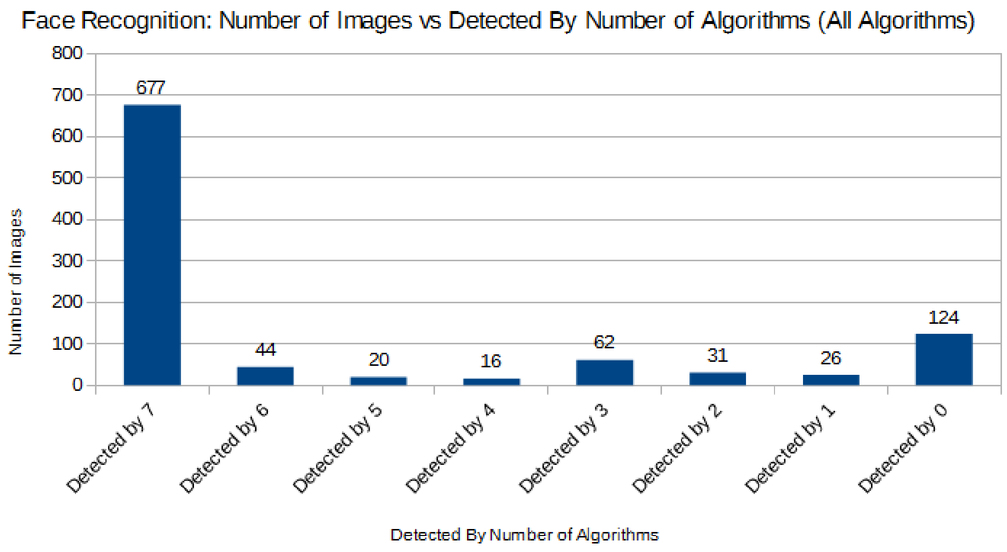


Figure 32: Face Recognition: Number of Images vs Detected by Number of Algorithms (All Algorithms)

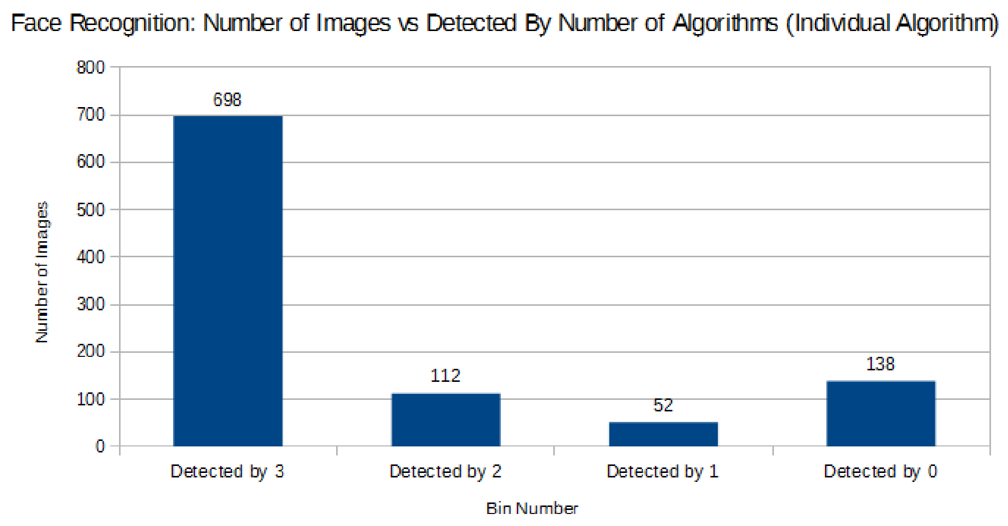


Figure 33: Face Recognition: Number of Images vs Detected by Number of Algorithms (Individual Algorithms)

The results show the cascaded algorithm combinations perform worse than their individual counterparts in facial detection. However, all face detection combinations still performed at an impressive 90% or higher accuracy rate. Cascaded approaches are therefore still relevant but do not offer significant enhancements to the overall success rate. The high success rate can be attributed to how each algorithm is best used for frontal faces. All of the faces used in the validation set are either frontal or various degrees of rotated frontal faces. Each algorithm does have some tolerance for fully profile faces (i.e. the Viola-Jones implementation has two different cascade profiles it looks at). Skin segmentation is shown to be an aggressive face detection algorithm. Individually, it performed the worst between the three individual algorithms. When a cascaded algorithm was used with skin segmentation, it suffered a significant drop in the number of faces detected.

FDR is seen to be small and lower than 1% across all combinations. All faces found by each combination was able to have the alignment process perform successfully on the cropped image representing the face. The FDR is slightly smaller for cascaded algorithms with the exception of HOG. This is ideal because it means images that were falsely detected as faces are fewer. The cascaded algorithms are then slightly more trustworthy than Viola-Jones and skin segmentation. By cascading the individual algorithms, more strict rules are imposed on the images on whether or not a certain part is considered a face. This factor is favorable if the image contained multiple objects in the background that one algorithm may detect incorrectly as a face. The subsequent algorithm in the cascade would then impose its own restrictions on the "face" and remove these invalid objects. Each algorithm looks at different aspects of a human face to determine eligibility. These differences can help filter out false positives. This project did not contain any images where there were various potential outlets for false positives. More data will help determine if the FDR will grow as the noise grows.

The execution times noted in Table 7 indicates how fast the algorithms performed. Because of the cascaded nature, the cascaded algorithms perform longer than their individual components. The amount of time however is not additive. Since each component in the cascade actually crops out the face image with some padding for subsequent cascades, each component will potentially look at a smaller image. This makes it the subsequent algorithm doesn't need to perform its analysis over the entire image. The smaller image is then ideally the face image where another pass is done over this section to verify it truly is a face. This is favorable since the FDR rate does decrease with multiple cascades and the additional rules imposed by another face detection algorithm may be desirable.

Despite being the most accurate face detection algorithm, HOG performs the slowest. Viola-Jones performs slightly better than HOG, but it does two passes where frontal and profile cascades are used on the image. If profile faces are not needed, then Viola-Jones can perform even faster. Both HOG and Viola-Jones have the tedious job of inspecting the image block by block looking for features. Skin segmentation is two orders of magnitude faster than either HOG or Viola-Jones. Despite all the optimizations that go into HOG and Viola-Jones like AdaBoost, cascade of features, or histograms; skin segmentation applies a skin mask threshold on the image. This implementation of skin segmentation does do additional operations like eroding and dilating, but it does not appear to consume a significant amount of time. This significant speed up cannot be ignored for face detection since the DSR is competitive to Viola-Jones to HOG. Since cascading adds an additional set of rules, skin segmentation can easily be added to form a cascade at a very cheap execution time cost. It should be noted that both HOG and Viola-Jones still operate in the milliseconds range, which is fast enough for many applications.

For facial recognition, all combinations' DSR and FDR are similar at around 81% and 19% as seen in Table 9. This means for the FEI face database, recognition is not affected by the cascaded nature. There is no compelling reason to use a cascaded vs an individual approach for frontal faces with no background noise. Despite HOG's high face detection rate, a large portion of those images cannot be recognized by the neural network. By using a full three component cascade, the number of faces examined by the neural network is fewer, suggesting these unrecognizable faces are filtered out in the face detection phase. This creates less work in the face recognition phase as the number of images that must be examined are smaller but still able to achieve the same success ratio as HOG individually. The performance of Viola-Jones and skin segmentation together is comparable to the performance of the three component cascade. This suggests that HOG can be removed and a two component cascade can be used with similar effectiveness with OpenFace's neural network model. The two HOG combinations with skin segmentation and Viola-Jones have similar performance with HOG's inherit high face detection rate and high miss face recognition rate.

The various algorithms are about 10% worst than OpenFace's accuracy when tested against the LFW benchmark. Using the same neural network model, OpenFace was able to achieve an accuracy rate of 92.92 ± 1.34%. The accuracy difference can be attributed to OpenFace's benchmark had the benefit of using a much larger training pool than the one used with this project. The number of items available in the training pool for this project was small with only 9 training images. With fewer training images, the classifier does not have sufficient data to classify a person with high accuracy.

Despite the mediocre results, the confidence score is low as seen in Table 10. This reflects the low amount of training images available per person. In this project, 9 images for 200 people were used as the training set for a total of 1800 images. This reflects the linear SVM was not able to create an ideal equation to represent the different face identities. This can be alleviated by increasing the number of training images for each person. Ideally the confidence scores are high for the correct identity and very low for all other identities. During the alignment phase, 173 images were discarded for a total of 1627 images used to train 200 identities. As seen in Figure 34, the number of training images per identity can be seen. Of the available training images, the roughly 90.4% of the training images were able to be aligned via dlib with only about 9.6% of the identities only having six or seven training images available. There were no identities that were training with five or less images.

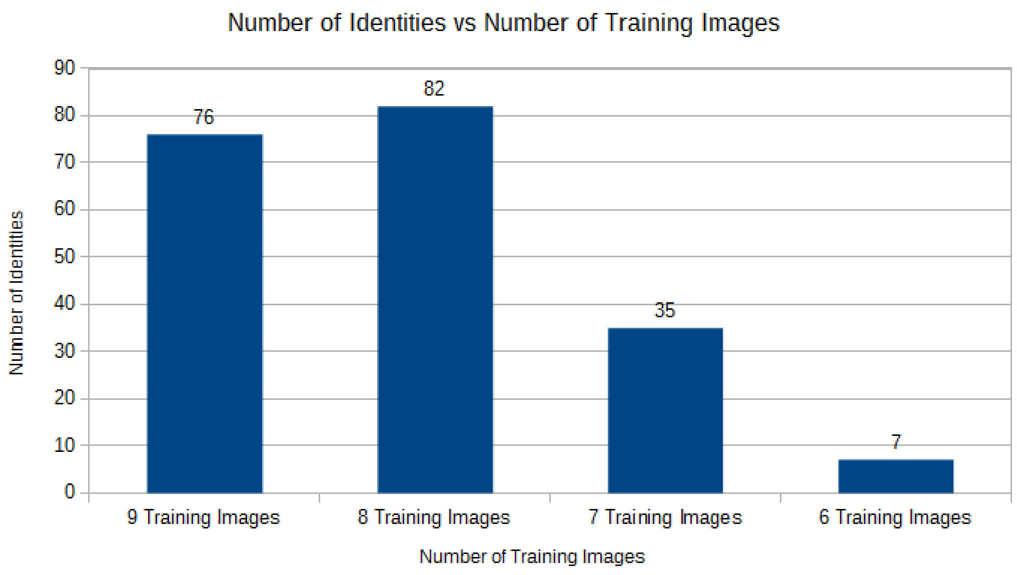


Figure 34: Number of Identities vs Number of Training Images

As seen in Table 10, the highest average confidence score is 5.3% for all algorithms. The drop off for the next entry is around 2.5% followed by around 1.6%. When the linear SVM makes an incorrect assertion on a face's identity, for all combinations, the average confidence score is about 3.7% and the subsequent score is about 2.6%. When it does make a correct assertion on a face's identity, the average confidence score jumps up to around 5.7% and the subsequent score is around 2.4%. It can be seen when incorrect assertions are made, the linear SVM is actually unsure about who the face belongs to. When incorrect, the difference between the highest and subsequent score is only 1.1%. When correct, the confidence difference between the highest and subsequent score is 3.3%; 3 times higher than the difference when it was wrong. In addition, the median for when the SVM is correct on who the face belongs to is about 5.5%. This indicates the majority of the time the SVM is right about the identity, it has a very high confidence of who the identity is. Potentially a threshold can be set at the median 5.5% for the confidence score to automatically filter out faces the SVM isn't confident about. This could be said to be unknown or unrecognizable faces. As seen in Figure 35 which describes the confidence score of the HOG, Viola-Jones, and skin segmentation combinations when facial recognition is correct, there are still some values that are in 3.7% range when the SVM makes an incorrect assertion.

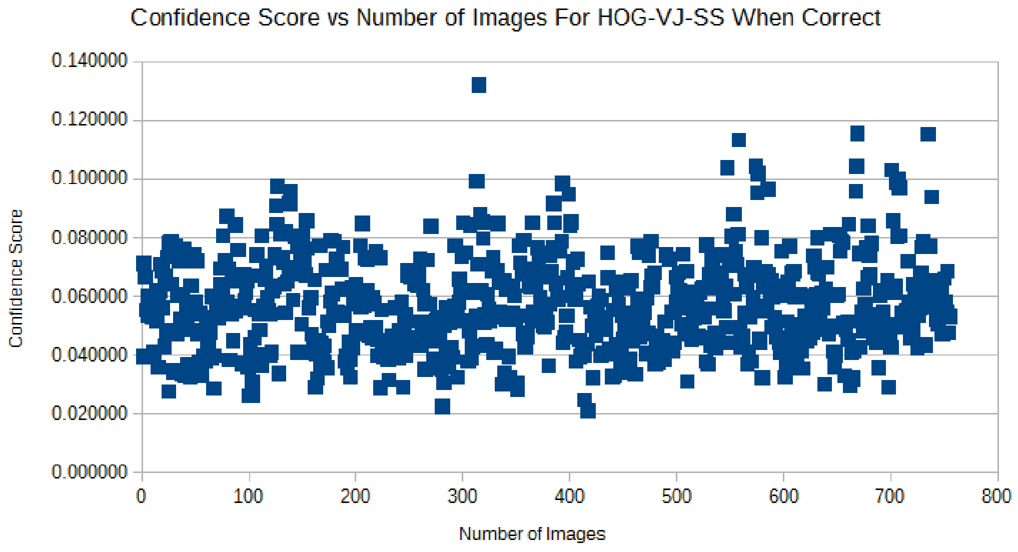


Figure 35: Confidence Score of HOG-VJ-SS When Correct

Setting the threshold to the incorrect assertion rate could introduce some false negatives but will preserve the majority of them. Table 11 shows various confidence thresholds for the HOG → Viola-Jones → Skin Segmentation cascade. The total number of correctly detected faces is 755. Consideration must be taken to select the optimal threshold score to not introduce too many false negatives. For this particular cascade, choosing a threshold score of 3.7% preserves a considerable amount of images while throwing away results with low confidence scores. This would lower the face recognition DSR to 74.5%, down from 81.5%, for this particular cascade combination.

Table 11: Face Recognition Threshold For HOG → Viola-Jones → Skin Segmentation

Threshold Confidence Score (%)	Total Faces Detected	Images Detected
2.4	755	753
2.6	755	752
3.7	755	690
5.5	755	384

Figure 30, 31, 32, and 32 shows how well the different algorithms agree with one another for facial detection or facial recognition. There were a total of seven different combinations like HOG, Viola-Jones, HOG → Viola-Jones → Skin Segmentation, and HOG → Viola-Jones. Figure 31 and 32 shows a subsection of Figure 30 and 32 in that they do not included the cascaded algorithms. These two charts show the individual components HOG, Viola-Jones, and skin segmentation. "Detected by 7" refers how how all combinations, cascaded and individual components, were able to detect n number of images. Subsequently, "Detected by 6" refers to how six combinations minus one unnamed algorithm was able to detect n number of images. Each column represents how many unique images were recognized by m number of algorithm. These charts show how well the FEI database performed against this test system. More images detected by all the combinations tested illustrates how each combination was able to agree with other combinations on certain images.

Figure 30 shows how well the seven combinations agreed with one another for face detection. The chart shows 921 unique images had the face detected among all seven algorithm combinations. Most images could be identified by all the face detection combinations. Enforcing stricter rules did not aggressively filter out face candidates in these images. Furthermore, there was no image that could not be detected by at least one combination. There is a minor group of images that could only be detected by three different algorithms. Upon further examination of this group, there does not seem to be any significant pattern to the type of image (i.e. orientation or expression) that failed. Each type of the five validation images had failed in this group with no particular image type failing especially miserably. A breakdown of what image types were located in this group can be seen in Table 12.

Table 12: Breakdown of Occurrences in "Detected By 3" for Face Detection

Image ID	Occurrence Count
03	9
04	9
07	10
08	16
13	7

Figure 31 shows the individual components break down for face detection. Since HOG achieved a detection rate of 100%, the "Detected by 2" column shows how many images were detected by both HOG and either Viola-Jones or skin segmentation. "Detected By 1" column shows 5 images were detected by HOG. "Detected By 0" has no images where a face could not be found. This shows the individual algorithms by themselves proved to be fairly robust in detecting faces in the images.

Figure 32 shows how well the seven combinations agreed with one another when a face was correctly detected. Each column represents how many images were recognized by the seven combinations. As seen in Figure 32, only 677 images were able to be correctly identified by all seven combinations. It should be noted that some images failed in certain combinations due to alignment issues. A total of 876 images out of the 1000 images could be correctly identified between all seven combinations. Given the maximum hit count was HOG which recognized 803 identities, 73 of these identities could be correctly identified by the other 6 combinations. This trend continues in the individual algorithms as seen in Figure 33. Between the three individual algorithms, a total of 862 images out of the 1000 images could be correctly identified. In this case, 59 images were recognized by Viola-Jones and skin segmentation that could not be recognized by HOG. Interestingly, this means some images were recognizable by the cascaded combinations but not recognizable by any of the individual algorithms. One example can be seen with identity 124 image label 3. The original image can be seen in Figure 36. The aligned training images can be seen in Figure 37. The resulting aligned image of HOG and HOG $\rightarrow$ Viola-Jones $\rightarrow$ skin segmentation can be seen in Figure 38. The HOG aligned image is on the left and the cascaded algorithm is on the right.

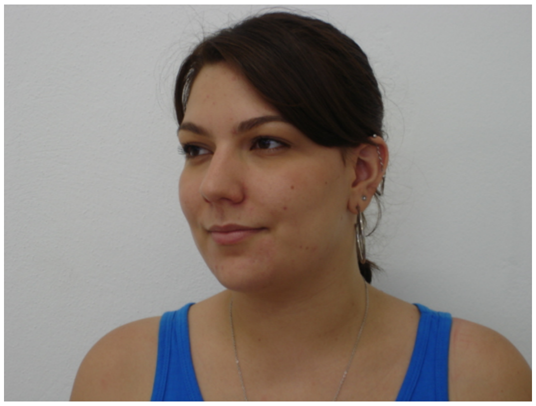


Figure 36: Identity 124 Image 03 Original Image.

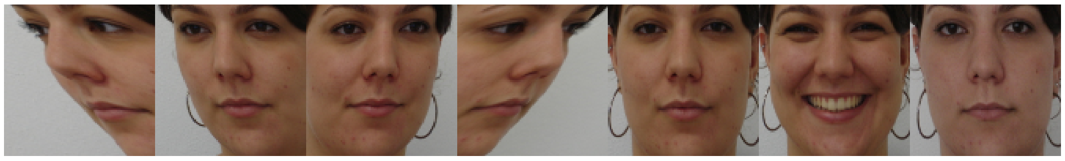


Figure 37: Identity 124 Image 03 Aligned Training Images.

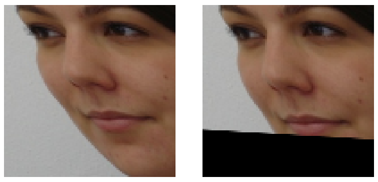


Figure 38: Identity 124 Image 03 HOG and HOG-VJ-SS Aligned.

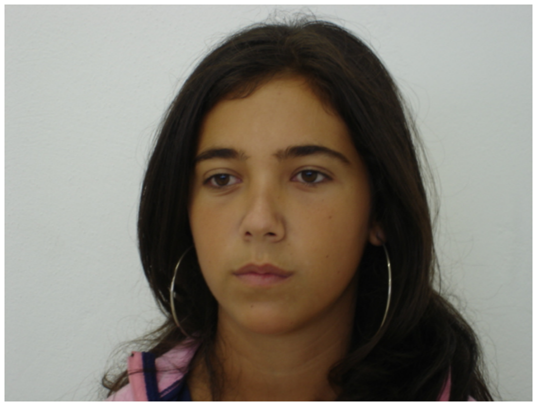


Figure 39: Identity 149 Image 05 Original Image.

Each of the individual algorithms was able to detect a face in identity 124 image 03, but incorrectly identified it with individual 149. Identity 149 can be seen in Figure 39. Skin segmentation had a confidence score of 2.9%. Viola-Jones had a confidence score of 3.7%. HOG had a confidence score of 4.1%. The three layer cascaded approach was able to correctly identify the face as identity 124 with a confidence score of 3.6%. All confidence scores are well below the median confidence score when a face is correctly identified. This indicates the aligned image for identity 124 label 03 are not reflective of the images in the training set. In the five validation images for identity 124, only two images were correctly identified by each individual algorithm. This number increases to three images correctly identified by the three layer cascaded algorithm. For all the incorrect identifications, the classifier mistook the face with other women in the FEI face database like identity 149. This suggests the features taken for identity 124 are not distinctive enough to differentiate it between other similar looking women. Interestingly when the neural network was able to correctly recognize the face, a portion of the chin is not visible.

10 Conclusion

In conclusion, the cascaded approach did not affect face detection or recognition by a very large margin for head shots with no background noise. The accuracy was similar for all the combinations where the cascaded implementations took longer than their individual counterparts. Cascaded approaches for frontal faces will not offer significant gains for face detection. The individual algorithms by themselves will perform with high enough accuracy. Each of the face detection schemes are quite adept at successfully finding faces in images. OpenFace's neural network proved to be very capable in determining who the face belonged to. This project's accuracy rate was roughly 10% lower due to the small amount of training images available. A strong improvement was not seen by implementing the three layer or two layer cascades, but the results do show that a cascaded algorithm has the same potential for correctly identifying faces with its stricter rules compared to the individual algorithms. Further testing needs to be done to see if these stricter rules benefit the cascaded algorithms when more noise is introduced in the image.

Cascaded approaches are not the best solution as seen by the execution times. The three layer and two layer cascades have comparative accuracies to the single HOG algorithm but require longer execution times. On the other hand, HOG was seen to perform the best at detecting faces. A combination of Viola-Jones and skin segmentation performs faster than HOG. If only frontal faces are looked at, Viola-Jones will have an even better speed up as the profile cascades are ignored. The recognition for the Viola-Jones and skin segmentation was slightly better than HOG. Therefore cascaded approaches should have the cascades selected based on the platform and the images used. Each individual algorithm offers its own benefits and rules that can help filter out false positives.

11 Future Work

Future work may include testing the cascaded model against more face databases with multiple images of the same person. Having more repeats of the same person will allow the classifier to be trained better to reduce the number of false positives. The face detection accuracy can be further improved by relaxing some of the constraints like the minimum neighbors parameter for Viola-Jones. For skin segmentation, another way to detect holes representing nostrils and mouths can be used to improve accuracy. The current method does not track where the holes are relative to each other. This could potentially help in identifying whether the holes detected are eyes relative to the parent's face.

Using alternative face databases with more frontal facing faces images can also help with the face detection algorithm. Doing tricks like warping the face to appear more frontal is always tricky and may not always give the best results. Another way to test the robustness of the cascaded approaches is to use images with full body humans, multiple objects in the background, or no humans.

Using a different neural network or classifier can give better results. There are multiple types of configurations one can configure a CNN with as well as train it with different data and iterations. Different classifiers may also group features differently than the linear SVM used and provide better and accurate confidences on who the person's face belongs to.

Unlike Viola-Jones, skin segmentation relies heavily on skin color. Viola-Jones discards this information as they are not necessary. However, skin color is an important factor in identifying the differences between people. The current YCbCr constraints appear to work fine with light skinned to dark skinned people. However, there may not be a uniform color threshold that works across all ethnicities. If it is known what type of skin color tones the face detector is going to be used on, it might be better to use a targeted color threshold rather than a general one. Using face database with certain ethnicities or skin colors can help identify the ideal threshold for each skin color tone.

To expand on the current idea, another implementation could be cascading the algorithms in different orders. Since the current method crops out the faces from the first cascade and feeds it into the second cascade, certain details are lost from the original image. If flipped around, skin segmentation may be more lenient on how big the face is compared to Viola-Jones. As seen in the results, there were certain cases where individually both algorithms agreed that a face existed but this result was not reflected in the proposed cascaded algorithm. Adding padding to the image could also help with preserving as much detail as possible while still maintaining the integrity of the face.

LIST OF REFERENCES

[1] F. Schroff, D. Kalenichenko, and J. Philbin, "Facenet: A unified embedding for face recognition and clustering," in *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2015, pp. 815–823.

[2] B. Amos, B. Ludwiczuk, and M. Satyanarayanan, "Openface: A general-purpose face recognition library with mobile applications," CMU-CS-16-118, CMU School of Computer Science, Tech. Rep., 2016.

[3] R. Mohanty and M. V. Raghunadh, "A new approach to face detection based on ycgcr color model and improved adaboost algorithm," in *2016 International Conference on Communication and Signal Processing (ICCSP)*, April 2016, pp. 1392–1396.

[4] M. M. Kasar, D. Bhattacharyya, and T. hoon Kim, "Face recognition using neural network: A review," *International Journal of Security and Its Applications*, vol. 10, no. 3, pp. 81–100, October 2016.

[5] A. Geitgey, "Machine learning is fun," 2014-2016. [Online]. Available: https://medium.com/@ageitgey/machine-learning-is-fun-80ea3ec3c471#.97rzyyv3b

[6] P. Viola and M. Jones, "Robust real-time object detection," in *International Journal of Computer Vision*, 2001.

[7] R. E. Schapire, *Explaining AdaBoost.* Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 37–52. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-41136-6_5

[8] O. H. Jensen, "Implementing the viola-jones face detection algorithm," 2008. [Online]. Available: https://pdfs.semanticscholar.org/40b1/0e330a5511a6a45f42c8b86da222504c717f.pdf?_ga=1.135568242.273406173.1489616969

[9] K. B. Shaik, P. Ganesan, V. Kalist, B. Sathish, and J. M. M. Jenitha, "Comparative study of skin color detection and segmentation in hsv and ycbcr color space," in *Procedia Computer Science*, 2015.

[10] H. Chang and U. Robles, "Face detection," May 2000. [Online]. Available: http://www-cs-students.stanford.edu/~robles/ee368/main.html

[11] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, vol. 1, June 2005, pp. 886–893 vol. 1.

[12] S. Mallick, "Histogram of oriented gradients," December 2016. [Online]. Available: http://www.learnopencv.com/histogram-of-oriented-gradients/

[13] C. McCormick, "Hog person detector tutorial," 2013. [Online]. Available: http://mccormickml.com/2013/05/09/hog-person-detector-tutorial/

[14] M. Nielsen, "Neural networks and deep learning," January 2016. [Online]. Available: http://neuralnetworksanddeeplearning.com/index.html

[15] karpathy@cs.stanford.edu, "Cs231n convolutional neural networks for visual recognition." [Online]. Available: http://cs231n.github.io/convolutional-networks/

[16] M. Davis and F. Sahin, "Hog feature human detection system," in *2016 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, Oct 2016, pp. 002 878–002 883.

[17] Y. Yang, C. Xie, L. Du, and Q. Lu, "A new face detection algorithm based on skin color segmentation," in *2015 Chinese Automation Congress (CAC)*, Nov 2015, pp. 523–526.

[18] S. Yadav and N. Nain, "Fast face detection based on skin segmentation and facial features," in *2015 11th International Conference on Signal-Image Technology Internet-Based Systems (SITIS)*, Nov 2015, pp. 663–668.

[19] G. B. Huang, M. Ramesh, T. Berg, and E. Learned-Miller, "Labeled faces in the wild: A database for studying face recognition in unconstrained environments," University of Massachusetts, Amherst, Tech. Rep. 07-49, October 2007. [Online]. Available: http://vis-www.cs.umass.edu/lfw

[20] L. Wolf, T. Hassner, and I. Maoz, "Face recognition in unconstrained videos with matched background similarity," Tech. Rep., 2011. [Online]. Available: https://www.cs.tau.ac.il/~wolf/ytfaces

[21] "Opencv (open source computer vision)." [Online]. Available: http://opencv.org

[22] R. Fisher, S. Perkins, A. Walker, and E. Wolfart, "Image processing learning resources hipr2," 2004. [Online]. Available: http://homepages.inf.ed.ac.uk/rbf/HIPR2/hipr_top.htm

[23] D. E. King, "Dlib-ml: A machine learning toolkit," *Journal of Machine Learning Research*, vol. 10, pp. 1755–1758, 2009.

[24] "torch: A scientific computing framework for luajit." [Online]. Available: http://torch.ch

[25] D. C. E. Thomaz, "Fei face database," March 2012. [Online]. Available: http://fei.edu.br/~cet/facedatabase.html

[26] P. Phillips, H. Wechsler, and P. R. J. Huang, “The feret database and evaluation procedure for face recognition algorithms,” pp. 295–306, March 2012.

[27] P. Phillips, H. Moon, S. Rizvi, and P. Rauss, “The feret evaluation methodology for face recognition algorithms,” pp. 1090–1104, March 2012.

[28] L. Buitinck, G. Louppe, M. Blondel, F. Pedregosa, A. Mueller, O. Grisel, V. Niculae, P. Prettenhofer, A. Gramfort, J. Grobler, R. Layton, J. VanderPlas, A. Joly, B. Holt, and G. Varoquaux, “API design for machine learning software: experiences from the scikit-learn project,” in *ECML PKDD Workshop: Languages for Data Mining and Machine Learning*, 2013, pp. 108–122.