Introduction to Visual Information Processing (CS455) Term Project

Face Detection & Eye Tracking

1) Author's Information

Ali Arda Eker

aeker1@binghamton.edu

2) Purpose

Implementing a face detection program using haar feature based cascade classifiers with OpenCV. Program detects the face then captures the eyes and irises in the face found. Then irises' middle point will be calculated and plotted on the screen. By this way program could tell which direction the user looks at horizontally and vertically in the later part of the project.

3) Applications

Road Safety:

A face detection software can be used to observe driver constantly during the travel in order to detect his or her face. Then program could check driver's mood to see if he is able to drive or not. It could sense if the driver is sleepy, anxious or exhausted by reading his face then warn him.

Also it can be used to detect driver's scanning ability. As we know drivers - especially the long road truck drivers – should watch the road constantly by scanning the vision from right to left and from top to bottom. They should check the rear view and inside mirrors all the time. A face detection program could be used to measure driver's ability to scan by reading his eye movements. It can also check if the driver looks his phone or somewhere else instead of the windshield.

Security:

Banks and other companies which need high security can use face detection programs to verify users' identity. A program could read the user's face or iris in order to match him or her in its database to see if the user has access to enter.

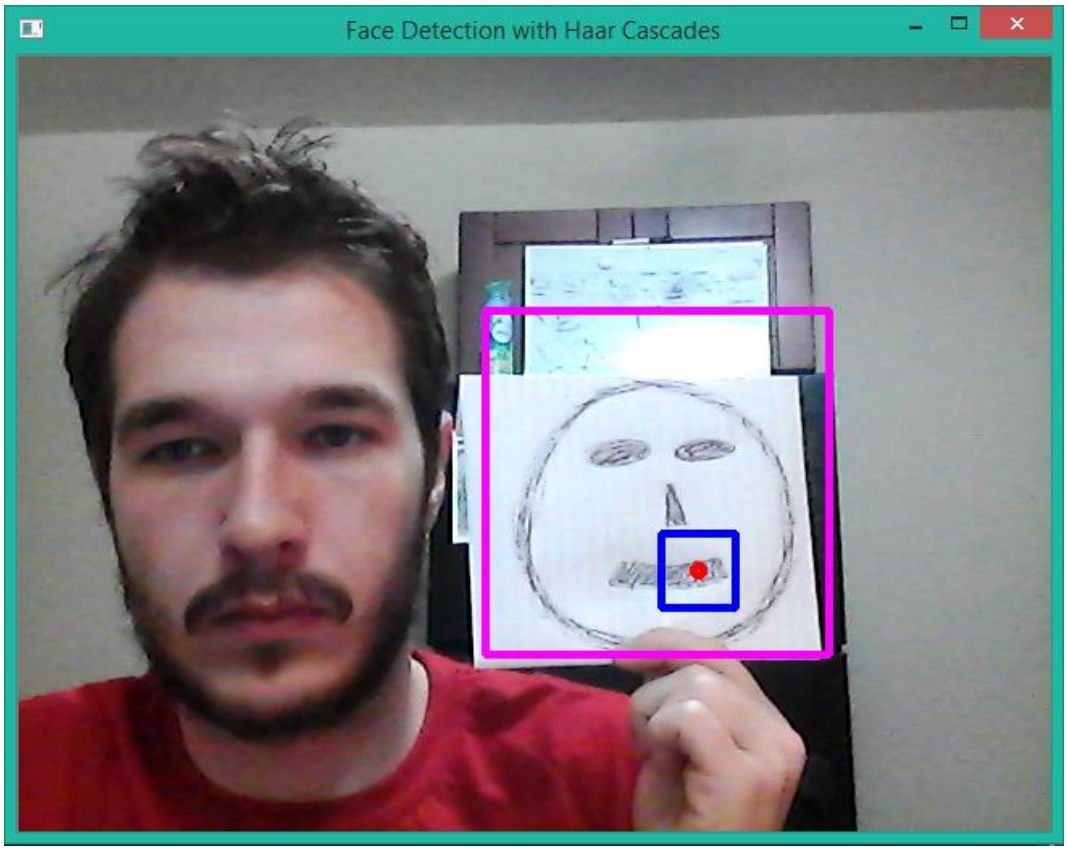
Healthcare Assistance:

A face detection program can be used to detect paralyzed patients' iris movements in order to help them to move their wheelchair. By that way patients who are not able to use a stick to move a wheelchair can control the direction of the vehicle just by looking at that way. With a camera attached to the wheelchair, we can sense where the patient looks at horizontally and vertically then move it to that way. My project will be used for this kind of applications.

4) Requirements

Program needs a plain and simple background without pictures or figures that can be detected as face. Since there will be a single patient who drives the wheelchair, program is implemented to detect just a single face so if the background contains any figures similar to a face, it can detect that instead of the patient's face as false positive.

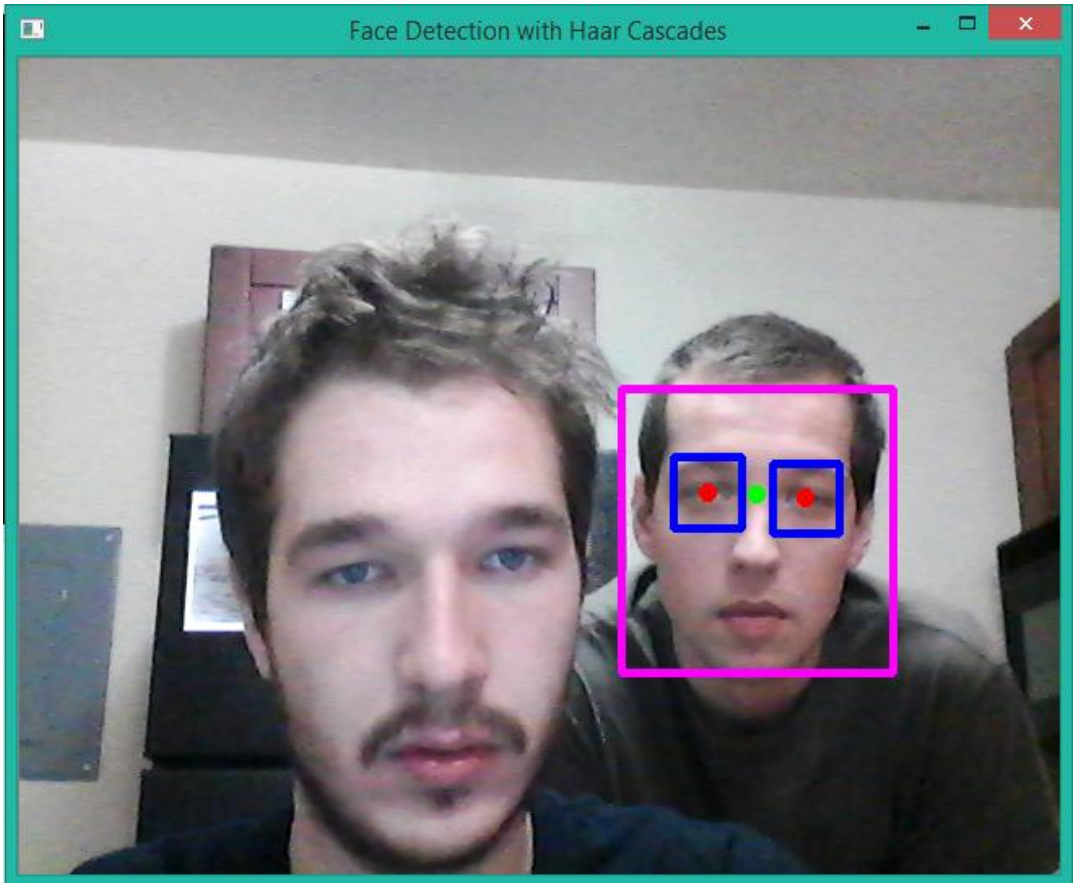
Example of a false positive detection as a figure similar to a face appears:



As it is seen above program captures a figure which is very similar to a face instead of my face. Since it is not a real face program could not detect the eyes or irises but it is enough to confuse the algorithm.

User should be alone in front of the camera for the same reason. If some other person walks behind the patient, program could capture that person's face instead of the patient's face as false positive.

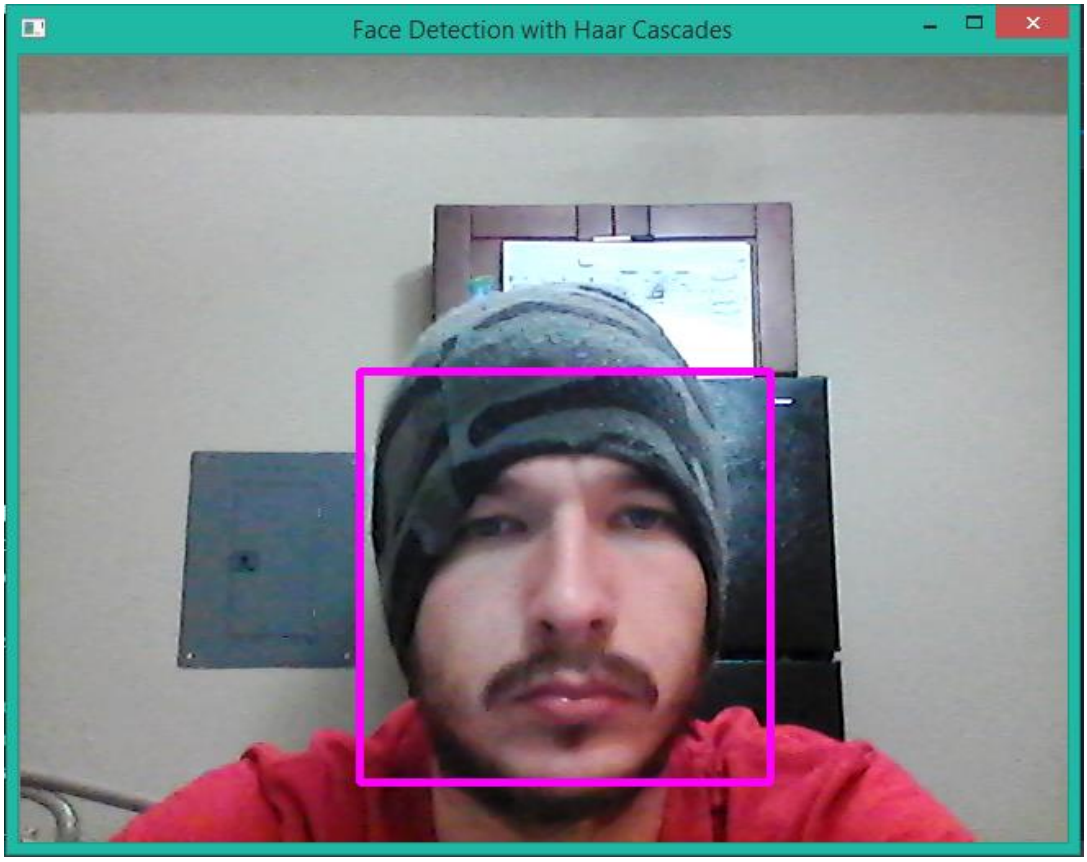
Example of a false positive detection as someone appears behind the user:



As it is shown on the above example, program could capture a face which does not belong to the user if someone appears behind him or her. Since the program is designed to detect only one face which should belong to the patient, when it detects another face, it automatically becomes unable to detect patient's face.

Program also expects a plain face from the user to work as designed. There should be no obstacles on the face of user that can alter the detection rate such as sun glasses, hat, scarf, beard or mustache.

Example of a false positive detection as the user carries an obstacle:



As it is seen in the above example, when the user wears a hat it can alter the process and modifies the detection rate. But although the user wears obstacles, the detection rate is still not too bad, program could capture the face. In this example it captures the face but could not detect the eyes so irises are not found. The program could still capture the eyes and irises because it is dependent some other factors too such as illumination, camera's type or clarity of the face. Thus this example is chosen among many outputs which are mostly true positive. And we can say that obstacles don't confuse the program as much as a second person or a figure in the background does.

5) Framework

The software is implemented with C++ using Visual Studio 2015 community edition in a 64 bit Windows machine. OpenCV library which is an open source computer vision framework is used for necessary functions and databases.

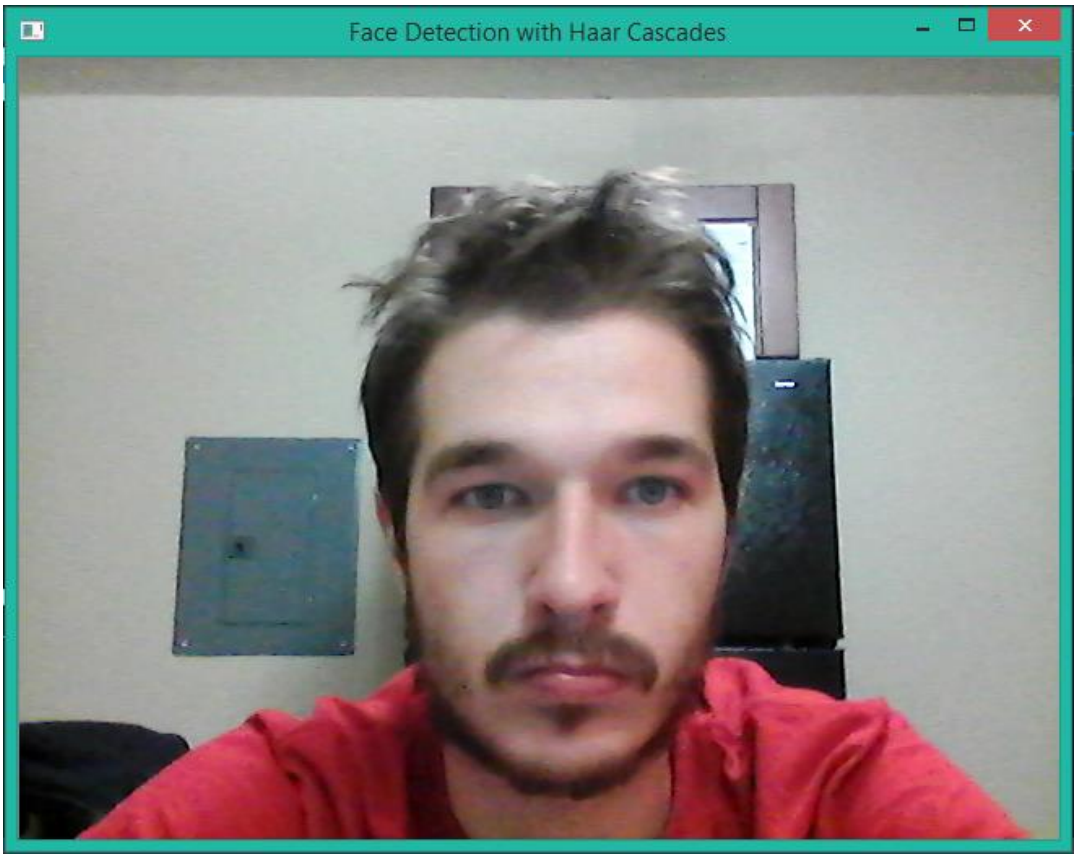
I used cascade features which are kept in the OpenCV xml files. I will talk about cascade features in detail later on the report. I also used OpenCV for camera operations, color converting, histogram equalization functions and Point, Rect, Mat structures.

6) Detection Rate

False Negative (FN):

The total number of faces that are not captured by the software divided by the total number of frames processed. It is the rate of faces that are forgotten. We want to minimize this rate.

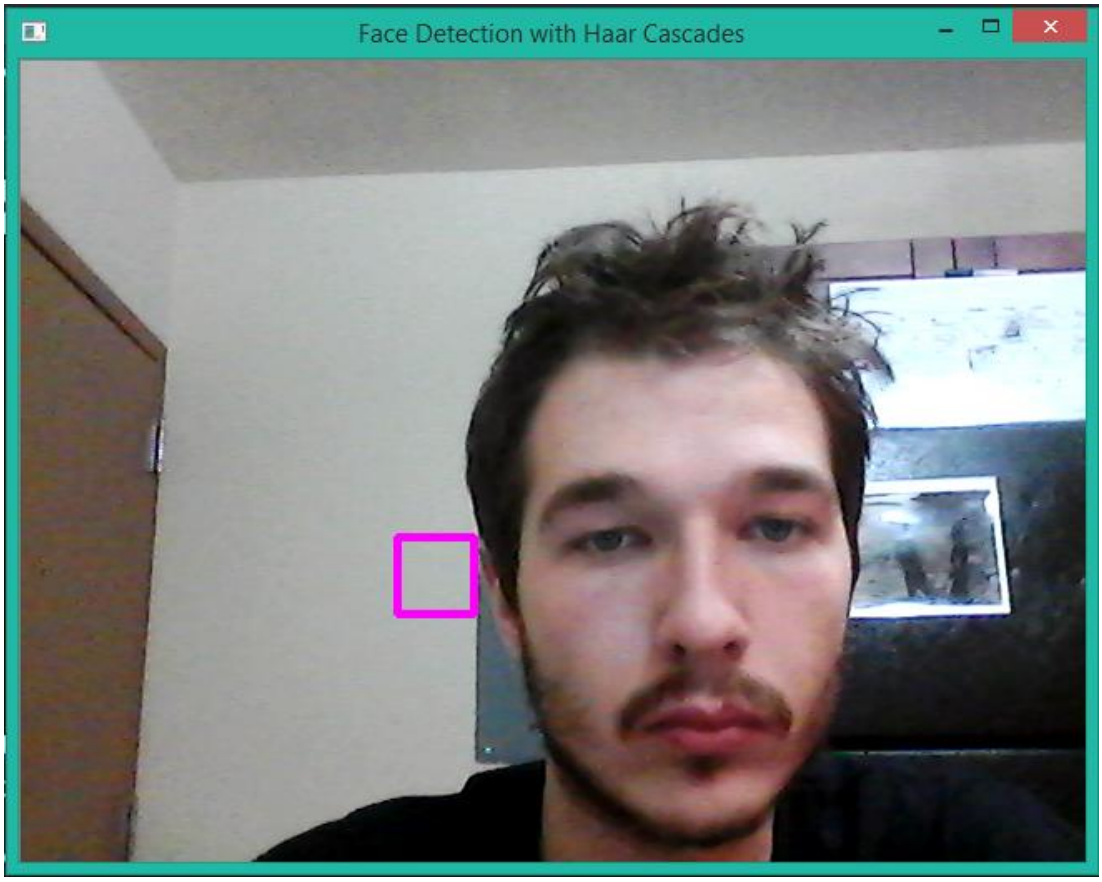
Example of a false negative output:



False Positive (FP):

The total number of non faces which are classified as face by the software divided by the total number of frames processed. It is the rate of wrongly detected faces. We want to keep this rate as low as possible.

Example of a false positive output:

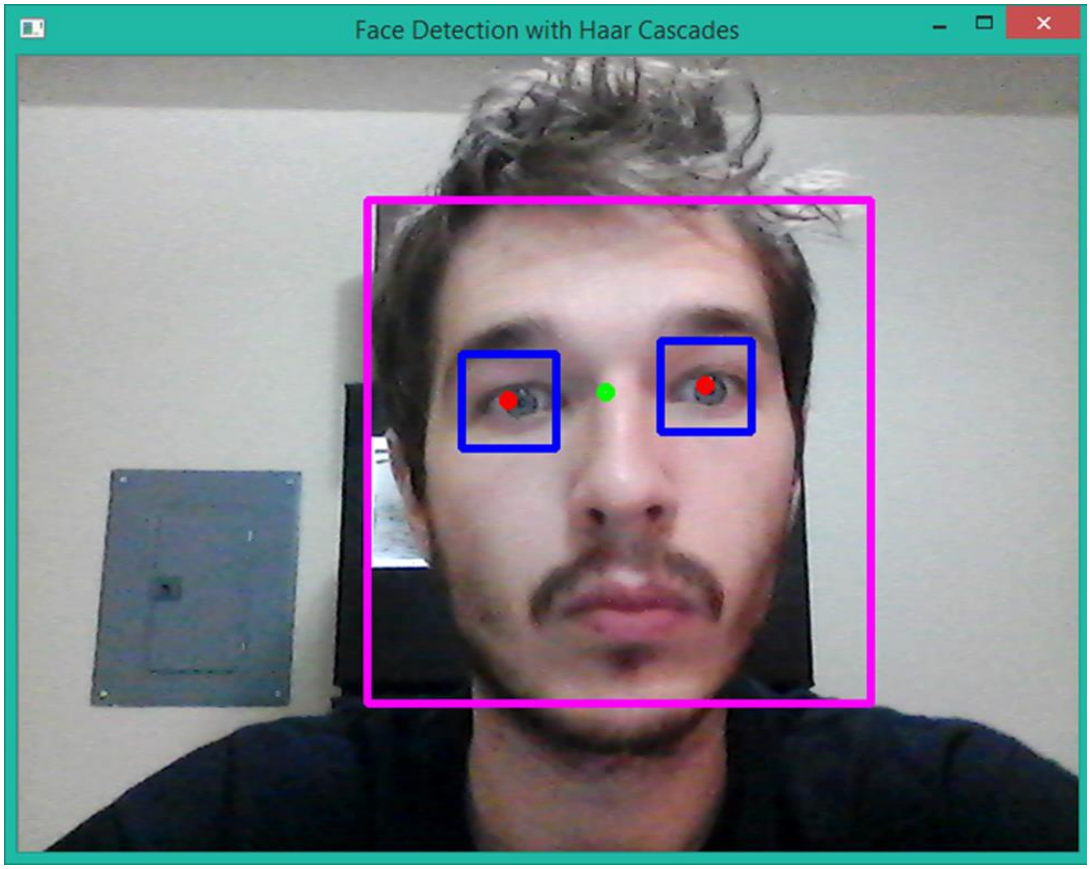


As it is shown above program could detect a non-face region as a face although the user does not carry any obstacles or the background is clear. This is a bug due to implementation and needs some more work on it. One of my solution ideas is I can keep the last captured face area in a variable during run time. Then I can compare every captured region's area with that variable. Since the false positive regions are always smaller than a face region as it is seen above I can discard those by checking the area difference of the last detected true positive face and the last detected region.

Detection Rate (d):

Detection rate in my program can be calculated as percentage of true positive faces captured per the total number of frames that are processed during run time. It can vary due to many factors which are obstacles carried by the user, background clarity, appearance of people behind the user, illumination, clarity of the face and camera's type. Although the user is in a perfect condition the program could still produce some false positive detections due to implementation bugs. We want to maximize this detection rate.

Example of a true positive output:



Above example shows an output with a truly detected face and eyes. It can be calculated by subtracting false negative rate from 1.

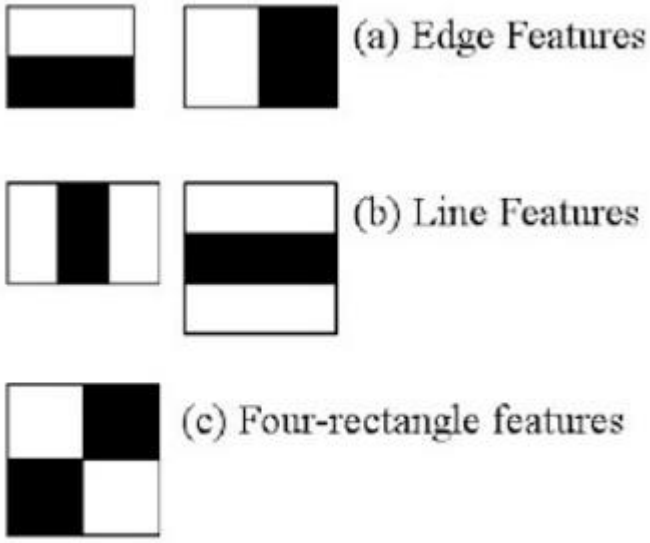
$D = 1 - FN$

7) Classification With Haar Feature Based Cascade Classifiers

Classification using haar feature based cascades is an effective object detection method proposed by Paul Viola and Michael Jones in their article, "Rapid Object Detection using a Boosted Cascade of Simple Features" in 2001.

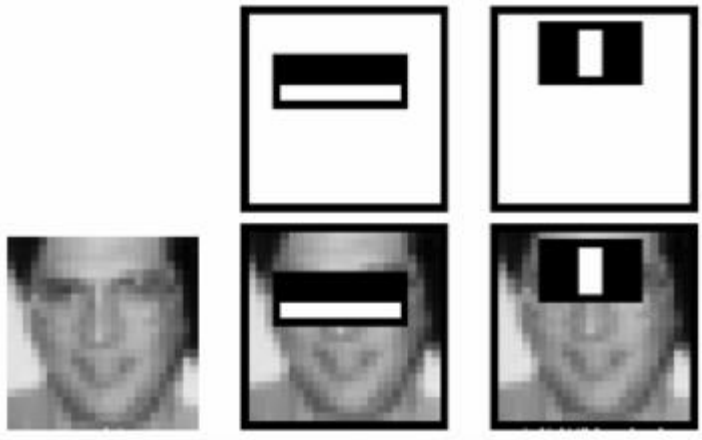
Although the algorithm can be used for any type of object detection, I used it for face and eye detection. The implementation of the algorithm is OpenCV's detectMultiScale function. It works as a method of cascade classifiers. In my program I used it as eyes_cascades.detectMultiScale() and faces_cascades.detectMultiScale(). I will talk about the cascades later in more detail. The function detects objects of different sizes in the input image. The detected objects are returned as a list of rectangle objects.

Initially the algorithm should have many positive and negative images to train the classifiers. For face detection positive inputs mean images with a face and negative inputs mean images without a face. Then haar features which are shown below are used to extract new features from trained classifiers. You can think them as convolutional kernels. Each feature represents a single value which is calculated by subtracting sum of pixels under white area from sum of pixels under black area.



Then all possible positions and locations of each kernel is used to extract as many features as possible. Now among these features calculated which can be a few hundred thousand, most of them are irrelevant. This means there can be features that are useless for detecting an object. For example below figure shows many possible features but the one on the far right is useless for detecting a face.

And below figure shows two good features. This means they are very useful for the program in order to detect a face. The feature on the left is good for us because it captures the property that the region of eyes is often darker than the region of the nose and cheeks. The feature on the right is also good because it captures the property that the eyes are darker than the bridge of the nose.



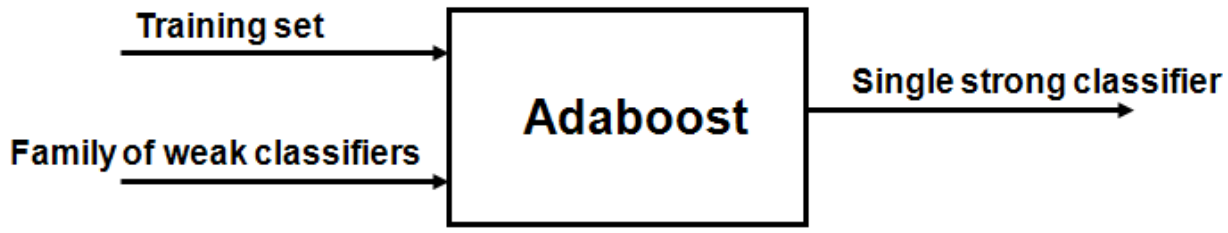
OpenCV contains many pre-trained classifiers such as faces, eyes or smiles.

8) Adaboost

As we have said previously that we can come up with a few hundred features because we would like to try each possible location and size for every kernel to extract the best ones for our application. So the question is how are we supposed to choose those best features among many? Here Adaboost algorithm takes place. It is used to discard irrelevant features and extract good ones.

Adaboost works like this: We apply each feature on all the training images. Then for each feature we find the best threshold value which will classify the images as positive or negative meaning images with a face or images without a face. During this process there happens to be misclassifications which mean some of the features indicate an image has a face but it does not or an image does not have a face but it does. After this we choose the features with minimum error rate which means they are the features that best classifies the face and non-face images.

Final classifier is a weighted sum of these weak classifiers. They are called weak because alone, they cannot classify the image correctly, but together with other weak classifiers they form a strong classifier which is able to classify images. Viola and Jones say that 200 features provide detection with 95% accuracy. We gained a reduction from a few hundred thousand to a few hundred.

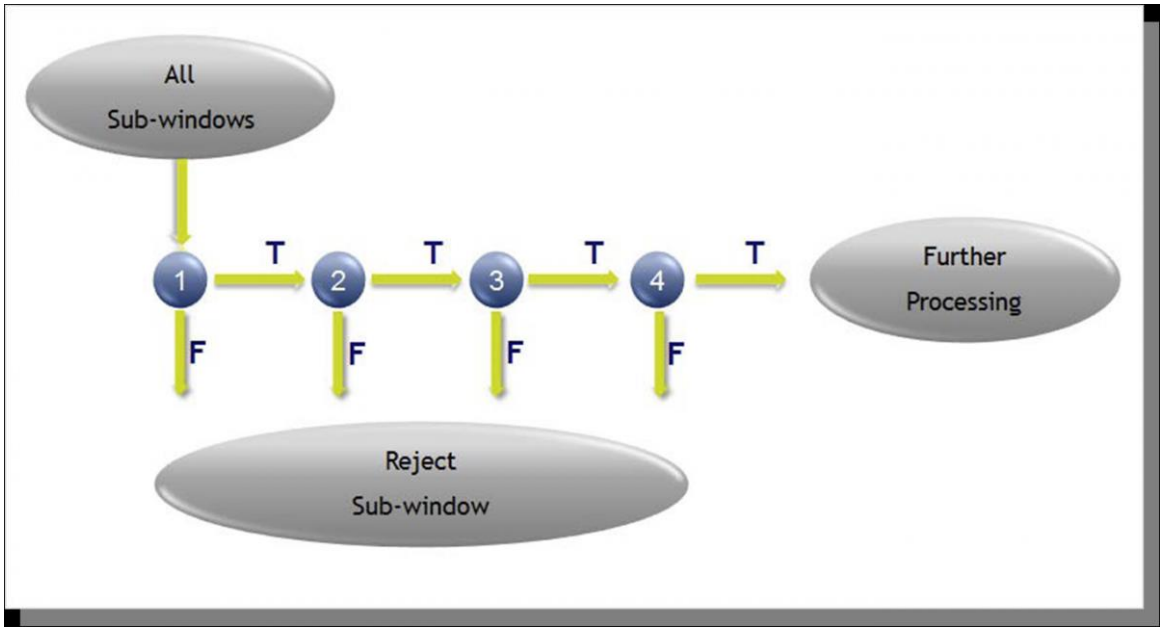


9) Cascade of Classifiers

We should apply previously chosen features which are proven to be useful to every window of each frame. If we take window size as 24x24 and if we have 200 good features, applying those to every frame of a video stream is very inefficient in terms of computational resources and it is time consuming. Thus Viola and Jones proposes the idea of cascade of classifiers.

In an image, we can say that most of the region is non face area. So it is better to check if a part of the image is a non-face region instead of checking if it is part of a face region. This saves us computationally because we can discard a part directly if it is a non-face part. Instead of focusing a part where there can be a face, we focus on parts which can be non-face area. This way, we can have more time to check a possible face region.

To do that Viola and Jones introduced the concept of Cascade of Classifiers. Instead of applying all of the features to every window of each frame, we group the features into different stages of classifiers and apply them one by one. First few stages contain a few number of features and as number of stages increase, they contain more. If a window fails we discard it directly. Since most of them fails in early stages, we gain resources by discarding them as early as possible. We don't consider remaining features on failed windows. If they pass, we apply the next stage of features and continue the process. The window which passes all stages is a face region.



Detector proposed on the article has more than 6K features with 38 stages of 1, 10, 25, 25 and 50 features. 2 features shown previously are the best ones from Adaboost. On average, 10 features out of 6K are evaluated per sub window.

10) Algorithm

The Program works like this, first of all we are getting the camera stream and parsing it into frames. Then we convert each frame from RGB to gray scale by using cvtColor function. After this step we equalize the histogram of each frame in order to normalize brightness and increase the contrast by using equalizeHist function.

Then first we detect the faces by using detectMultiScale function and face cascades and second we detect the eyes within the face just found using detectMultiScale with eyes cascades. Finally we calculate the position of irises with their central point and plot them.

11) Conclusion & Discussion

As we have discussed how face detection is performed using Haar feature based cascade classifiers and Adaboost algorithm, we stated the requirements for achieving best results and obstacles for the program. Future work will be this, I will develop the program further to measure the change of central point of the irises in order to calculate where user looks at horizontally and vertically. Then this will be used to move a wheelchair who are driven by paralyzed patients.

12) References

[1] P. Viola and M. Jones. *Rapid Object Detection using a Boosted Cascade of Simple Features*, 2001.

[2] J. Meynet. *Fast Face Detection Using AdaBoost*, 2003.