ORIENTAL INSTITUTE OF SCIENCE & TECHNOLOGY

Real Time Face Detection

Using KLT &Viola Jones Algorithm

A

Project Work

Submitted as Major Project in Partial fulfillment for the award of Graduate Degree in Bachelor of Technology in Computer Science & Engineering.

Submitted to

RAJIV GANDHI PROUDYOGIKI VISHWAVIDYALAYA
BHOPAL (M.P)

Submitted By--

Charchit Somankar(0105CS171034)
Iram Khan(0105CS171044)
Nairitya Tale(0105CS171062)

Under the Guidance of

Prof. Manjari Singh Rathore
(Department of Computer Science & Engineering)

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

JUL-DEC 2020

ORIENTAL INSTITUTE OF SCIENCE & TECHNOLOGY

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

JUL-DEC 2020

CERTIFICATE

This is to certify that the project entitled ***Real Time Face Detection"*** *being submitted by* ***Charchit Somankar, Iram Khan, and Nairitya Tale,*** *students of V//th Semester, Degree in Computer Science & Engineering have done their work as MAJOR PROJECT for Partial fulfillment of the degree from RGPV, Bhopal (M.P.) is a record of bonafide work carried out by them under our supervision.*

Prof. Manjari Singh Rathore
Guide
Department of
Computer Science & Engineering

Prof. Farha Haneef
Head
Department of
Computer Science & Engineering

ACKNOWLEDGEMENT

I take the opportunity to express my cordial gratitude and deep sense of indebtedness to my guide **Prof. Manjari Singh Rathore** for the valuable guidance and inspiration throughout the project duration. I feel thankful to her for her innovative ideas, which led to successful submission of this major project work. I feel proud and fortune to work under such an outstanding mentor in the field of **Machine learning**. She has always welcomed my problem and helped me to clear my doubt. I will always be grateful to her for providing me moral support and sufficient time.

I owe sincere thanks to **Dr. *K.G. Sudhakar*,** Director OIST, for providing me with moral support and necessary help during my project work in the Department.

At the same time, I would like to thank **Dr. Farha Haneef (HOD, CSE)** and all other faculty members and all non-teaching staff of department of Computer Science & Engineering for their valuable co-operation.

I would also thank to my Institution, faculty members and staff without whom this project would have been a distant reality. I also extend my heartfelt thanks to my family and well wishers.

Charchit Somankar
(0105CS171034)

Iram Khan
(0105CS171044)

Nairitya Tale
(0105CS171062)

APPROVAL CERTIFICATE

This is to certify that the project entitled **Real time face detection** being submitted by Charchit Somankar (0105CS171034), Iram Khan (0105CS171044), Nairitya Tale (0105CS171062) students of Eighth Semester, Bachelor of Engineering in Computer Science and Engineering have done their work as Major Project for Partial fulfillment of the degree from RGPV, Bhopal (M.P.).

Guide Name: Prof. Manjari Singh Rathore

Date: 11/01/2021

Signature

Abstract

Face detection by computer systems has become a major field of interest. Face detection algorithms are used in a wide range of applications, such as security control, video retrieving, biometric signal processing, human computer interface, face recognitions and image database management. However, it is difficult to develop a complete robust face detector due to various light conditions, face sizes, face orientations, background and skin colors.

In this report, we use two approaches for detecting a face and track it continuously. Basically video sequences provide more information than a still image. It is always a challenging task to track a target object in a live video. We undergo challenges like illumination; pose variation and occlusion in pre-processing stages. But this is can be overcome by detection of the target object continuously in each and every frame. Face tracking by Kanade Lucas Tomasi algorithm that is used to track face based on trained features. Whereas the Viola Jones algorithm is used detect the face based on the haar features.

INDEX

CONTENTS PAGE

1. INTRODUCTION:

A face detection system is a computer application for automatically detecting human face from a digital image or a video frame from a video source. Face detection is a pre-processing of face recognition. It is also used for the security system.

1.1 Overview

The face is our primary focus of attention in social life playing an important role in conveying identity and emotions. We can recognize a number of faces learned throughout our lifespan and identify faces at a glance even after years of separation. This skill is quite robust despite of large variations in visual stimulus due to changing condition, aging and distractions such as beard, glasses or changes in hairstyle.

Human-robot interaction receives an increasing attention among the researches in different areas of interest during last years. In the current project we address one of the computer vision tasks, involved in developing of such an interactive system. Namely, we are interested in detecting human user presence and tracking of his attention.

Detecting human faces in a video is a great challenging problem. These configurations may be like angle of view, background intensity, and various illuminations. This is due to high variety of configurations that may occur. The complexness of the face results in a particular degree of issue for fast detection and tracking.

For face detection and tracking in a given video sequence different algorithms have been introduced over the past few years. Each algorithm has got its own advantages and disadvantages. But any face tracking algorithm will have some errors which will cause

deviation from the required object. The tracker can be accurate if and only if it is able to minimize this deviation. The technique used in this report is one of the effective approaches. It is quicker and simpler and we make use of the eigen vectors for detecting the faces along with the superficial points of our faces.

Object detection and tracking are important in many computer vision applications including activity recognition, automotive safety, and surveillance. In this example, you will develop a simple face tracking system by dividing the tracking problem into three parts:

1. Detect a face
2. Facial features to track
3. Track the face
4. No. of faces.

2.PROJECT OBJECTIVE AND SCOPE

2.1 Project Objectives:

There are few objectives to design face detection system. The objective of face detection are :

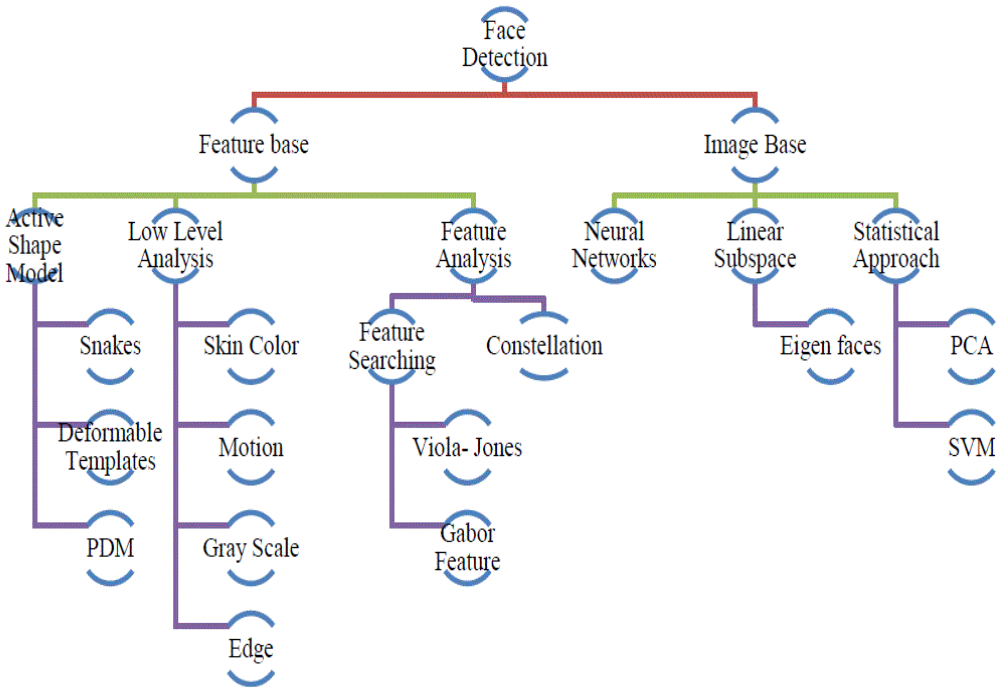
- To design real time face detection system.
- To utilize the face detection system based on Haar Classifier.

2.2 Project Scope:

- The first scope of this project is to develop a face detection based on Haar Classifier. Haar Classifier is used because it achieved high detection accuracy. Haar Classifier can efficiently reduce or increase the class variability and making the classification easier.
- The second scope is to extract the human face. It will extract the desire image, for this system it will extract human face using Haar Classifier.

3. LITERATURE SURVEY

Face detection is a computer technology that determines the location and size of human face in arbitrary (digital) image. The facial features are detected and any other objects like trees, buildings and bodies etc are ignored from the digital image. It can be regarded as a specific case of object-class detection, where the task is finding the location and sizes of all objects in an image that belong to a given class. Face detection, can be regarded as a more general case of face localization. In face localization, the task is to find the locations and sizes of a known number of faces (usually one). Basically there are two types of approaches to detect facial part in the given image i.e. feature base and image base approach. Feature base approach tries to extract features of the image and match it against the knowledge of the face features. While image base approach tries to get best match between training and testing images.



3.1 Face Tracking

Object tracking is defined as keeping a trace on a particular kind of object. In this paper as we are mainly concentrating on face, we track human faces based on the given input features. Continuous tracking makes us leave the problems like illumination, variation in pose etc. aside . Here tracking of human faces in a video sequence is done and also live video tracking using a webcam is done.

3.2 Kanade Lucas Tomasi (KLT) algorithm

Kanade Lucas Tomasi algorithm is used for feature tracking. It is a most popular one. KLT algorithm was introduced by Lucas and Kanade and their work was later extended by Tomasi and Kanade. This algorithm is used for detecting scattered feature points which have enough texture for tracking the required points in a good standard .

Kanade-Lucas-Tomasi (KLT) algorithm is used here for tracking human faces continuously in a video frame. This method is accomplished by them finding the parameters that allow the reduction in dissimilarity measurements between feature points that are related to original translational model.

Firstly in this algorithm we calculate the displacement of the tracked points from one frame to another frame. From this displacement calculation it is easy to compute the movement of the head. The feature points of a human face are tracked by using optical flow tracker [4]. KLT tracking algorithm tracks the face in two simple steps, firstly it finds the traceable feature points in the first frame and then tracks the detected features in the succeeding frames by using the calculated displacement.

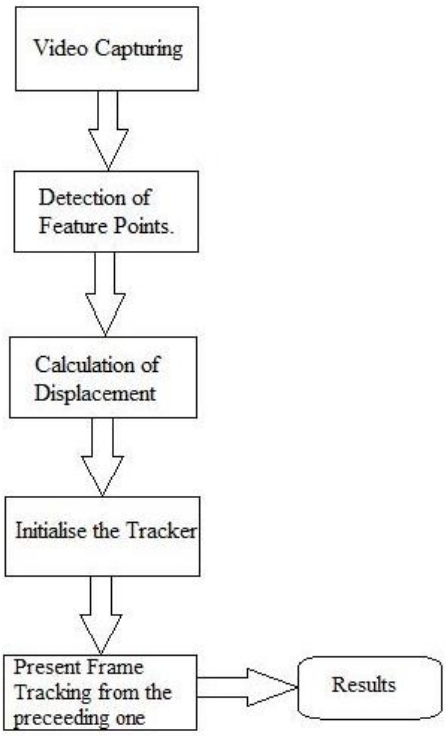


Figure 2.1: KLT Algorithm

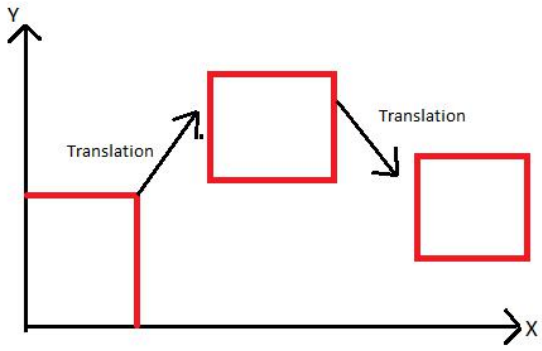


Figure 2.2: Translation

In this algorithm firstly it detects harris corners in the first frame. And then using optical flow it continues to detect the points by computing the motion of the pixels of an image. For each translation motion of the image optical flow is computed. The harris corners are detected by linking the motion vectors in successive frames to get a track for each harris point. Just not to lose the track of the video sequence we apply harris detector at every 10 to 15 frames. This is nothing but making sure by checking the frames periodically. In this way new and old harris points are tracked. Here in this paper we consider only 2-D motion i.e. translation movement.

Let us assume that the initially one of the corner point is (x, y). Then in the next frame, if it is displaced by some variable vector (b1, b2,.bn), the displaced corner point of the frame will be the sum of the initial point and displaced vector. The coordinates of the new point will be x=x+b1 and y=y+b2. So, the displacement now should be calculated with respect to each

coordinate. For this we use warp function which is a function with coordinates and a parameter. It is denoted as W(x ;p) =

(x+b1; x+b2). The warp function is used to estimate the formation.

3.2.1 Alignment Estimation

In the first frame the initially detected points is taken as a template image. In the later stages difference between displacement and the preceding point is taken to get the next tracking points. The alignment is calculated by an expression and later it is calculated using Taylor Series. Where p is the displacement parameter Assume initial estimate of p as a known parameter, where H is called as hessian matrix. This is how we estimate the displacement and find the next traceable point . This is beyond the scope of our research. We consider it as a future scope of our current work.

3.3 Face Detection

Detecting a face is a computer technology which let us know the locations and sizes of human faces. This helps in getting the facial features and avoiding other objects and things. In the present situation human face perception is a biggest research area. It is basically about detecting a human face through some trained features. Here face detection is preliminary step for many other applications such as face recognition, video surveillance etc.

The face detection system can be divided into the following steps :-

1. **Pre-Processing:** To reduce the variability in the faces, the images are processed before they are fed into the network. All positive examples that is the face images are obtained by images with frontal faces to include only the front view. All the cropped images are then corrected for lighting through standard algorithms.

2. **Classification:** Neural networks are implemented to classify the images as faces or nonfaces by training on these examples. We use both our implementation of the neural network and the Matlab neural network toolbox for this task. Different network configurations are experimented with to optimize the results.

3. **Localization:** The trained neural network is then used to search for faces in an image and if present localize them in a bounding box. Various Feature of Face on which the work has done on:- Position Scale Orientation .

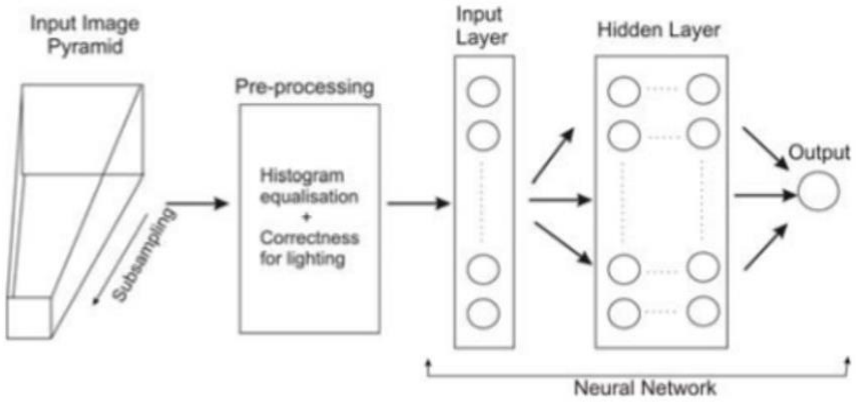


Fig: Face detection algorithm

3.4 Viola-Jones Algorithm

This algorithm helps us detect features of a face in a particular frame of a video sequence. This is the first object detection framework which gives a competition to real time detection rates. Paul Viola and Michael Jones are the ones who introduced this algorithm. They made this algorithm mainly by the issue of face detection. There are four steps which have to be followed to detect a face. Firstly, we train the system with the haar features [6-8]. Haar features are a kind of rectangular boxes which are black and white.

3.4.1 Haar Features

Haar features are simple rectangular feature which is the difference of the sum of pixels of areas inside the rectangle. This rectangle can be at any position of the frame and can scale the image. This modified feature set is called 2-rectangle feature. Each feature type can indicate the existence or the absence of certain characteristics in the frame, such as edges or changes in texture. These haar features are applied to determine the facial features. The Black part is used to detect nose feature of a human face as the black colored part defines the presence of a nose which is located at the center of the face. And the Figure-3 (e) is called a 4 rectangle feature. Where the black part is denoted as +1 and the white part is denoted as -1. The result is calculated by subtracting the sum of pixels under the white rectangle from the sum of pixels

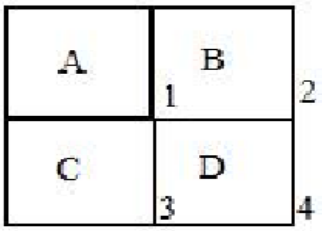


Figure 2.3: Integral Image Calculation

under black rectangle. Initially some threshold is taken for particular features. Average sum of each black and white is calculated. Then difference is checked with threshold. If the value is above or matches with the threshold then it is detected asrelevant feature.

3.4.2 Integral Image Test

The integral image part is used to sum all the pixels of a particular box to its left and above ones. The four corner values of the area are to be calculated. This makes avoid summing of each pixel in the region. This integral image conversion process is introduced just to speed up the process in calculating pixels.

The calculation of the sum of pixels of part D in the fig.4 is (1+4) (2+3) i.e. [A+(A+B+C+D)] - [(A+B+A+C)] which gives D. The authors defined the base resolution of the detector

to be 24x24. In other words, every image frame should be divided into 24x24 sub windows, and features are extracted at all possible locations and scales for each such sub-window. This

results in an exhaustive set of rectangle features which counts more than 160,000 features for a single sub-window.

3.4.3 Adaboost

It is a process used to find out relevant and irrelevant features. It uses the weak classifiers and weights to form a strong classifier. It finds the single rectangular feature and threshold which is the best to separate the faces and non-faces in training examples in terms of weighted error. It firstly starts with uniform weights while training. Next it evaluates the weighted error for each feature and picks the best. We reevaluate the examples where incorrect classifiers will have more weight and correct classifiers will have less weight. Finally the classifier will contain the combination of correct classifiers which are having less weight. To reduce the computational time non-faces are discarded.

3.4.4 Learning Classification Functions

The complete set of features is quite large - 160,000 features per a single 24x24 sub-window. Though computing a single feature can be done with only a few simple operations, evaluating the entire set of features is still extremely expensive, and cannot be performed by a real-time application. Viola and Jones assumed that a very small number of the extracted features can be used to form an effective classifier for face detection. Thus, the main challenge was to find these distinctive features.

They decided to use AdaBoost learning algorithm as a feature selection mechanism.

In its original form, AdaBoost is used to improve classification results of a learning algorithm by combining a collection of weak classifiers to form a strong classifier. The algorithm starts with equal weights for all examples. In each round, the weight are updated so that the misclassified examples receive more weight. By drawing an analogy between weak classifiers and features, Viola and Jones decided to use AdaBoost algorithm for aggressive selection of a small number of good features, which nevertheless have significant variety. Practically, the weak learning algorithm was restricted to the set of classification functions, which of each was dependent on a single feature. A weak classifier h(x, f, p,) was then defined for a sample x (i.e. 24x24 sub-window) by a feature f, a threshold , and a polarity p indicating the direction of the inequality. The key advantage of the AdaBoost over its competitors is the speed of learning. For each feature, the examples are sorted based on a feature value. The optimal threshold for that feature can be then computed in a single pass over this sorted list.

3.4.5 Cascading

This step is introduced to speed up the process and give an accurate result. This step consists of several stages where each stage consists of a strong classifier. All features are grouped into several stages. It detects faces in the frame by sliding a window over a frame. When an input is given it checks for certain classifier in the first stage and then so on. But it is passed to the successive stage if and only if it satisfies the preceding stage classifier.

4. PROPOSED METHOD

4.1 Algorithm

We have proposed one modified algorithm which uses the underlying principles of KLT Algorithm and Viola-Jones Algorithm, but it works somehow better than that and it has been proved. We have used the concept of finding the eigen vectors for live stream and this amount of substantial work hasn't been done before for real time systems, but for video stream, research has been done.

1. Detect the Face- First, you must detect the face. Use the vision. Cascade Object Detector System object to detect the location of a face in a video frame. The cascade object detector uses the Viola-Jones detection algorithm and a trained classification model for detection. By default, the detector is configured to detect faces, but it can be used to detect other types of objects.

To track the face over time, our implementation uses the Kanade-Lucas-Tomasi (KLT) algorithm. While it is possible to use the cascade object detector on every frame, it is computationally expensive. It may also fail to detect the face, when the subject turns or tilts his head. This limitation comes from the type of trained classification model used for detection. Our work detects the face only once, and then the KLT algorithm tracks the face across the video frames.

2. Identify Facial Features to Track- The KLT algorithm tracks a set of feature points across the video frames. Once the detection locates the face, it identifies feature points that can be reliably tracked.

3. Initialize a Tracker to Track the Points - With the feature points identified, you can now use the vision. Point Tracker System object to track them. For each point in the previous frame, the point tracker attempts to find the corresponding point in the current frame. Then the estimate Geometric Transform function is used to estimate the translation, rotation, and scale between the old points and the new points. This transformation is applied to the bounding box around the face.

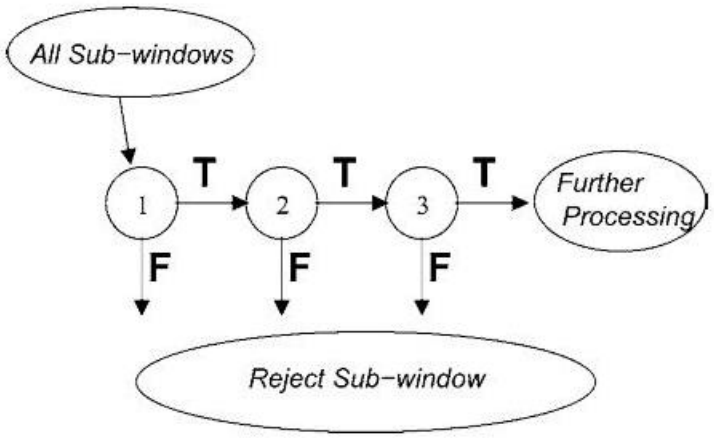


Figure 3.1: Detectors Cascade (See Below)

4. Initialize a Video Player to Display the Results

5. Track the Face

4.2 Detectors Cascade

There is a natural trade-off between classifier performance in terms of detection rates and its complexity, i.e. an amount of time required to compute the classification result. Viola and Jones , however, were looking for a method to speed up performance without compromising quality.

As a result, they came up with an ideaof detectors cascade (see Figure 4). Each sub-window is processed by a series of detectors, called cascade, in the following way. Classifiers are combined sequentially in the order of their complexity, from the simplest to the most complex. The processing of a sub-window starts, then, from a simple classifier, which was trained to reject most of negative (non-face) frames, while keeping almost all positive (face) frames. A sub-window proceeds to the following, more complex, classifier only if it was classified as positive at the preceding stage. If any one of classifiers in a cascade rejects a frame, it is thrown away, and a system proceeds to the next sub-window. If a sub-window is classified as positive by all the classifiers in the cascade, it is declared as containing a face.

Since most of sub-windows do not contain faces, only a small number of them are processed by all the classifiers in a cascade. The majority of subwindows are rejected during first few stages, involving simple classifiers solely, which results can be evaluated very rapidly. Thereby, the overall detection performance meets the requirements of real-time processing, and yet it yields high detection rates, comparable with alternative, much slower, techniques.

4.3 Viola Jones Method:

Paul Viola and Michael Jones presented an approach for object detection which minimizes computation time while achieving high detection accuracy. Paul Viola and Michael Jones [39] proposed a fast and robust method for face detection which is 15 times quicker than any technique at the time of release with 95% accuracy at around 17 fps. The technique relies on the use of simple Haar-like features that are evaluated quickly through the use of a new image representation. Based on the concept of an ―Integral Image‖ it generates a large set of features and uses the boosting algorithm AdaBoost to reduce the overcomplete set and the introduction

of a degenerative tree of the boosted classifiers provides for robust and fast interferences. The detector is applied in a scanning fashion and used on gray-scale images, the scanned window that is applied can also be scaled, as well as the features evaluated.

The Viola-Jones algorithm is a widely used mechanism for object detection. The main property of this algorithm is that training is slow, but detection is fast. This algorithm uses Haar basis feature filters, so it does not use multiplications.

The efficiency of the Viola-Jones algorithm can be significantly increased by first generating the integral image.

$$II(y,x) = \sum_{p=0}^{y} \sum_{q=0}^{x} Y(p,q)$$

The integral image allows integrals for the Haar extractors to be calculated by adding only four numbers. For example, the image integral of area ABCD (Fig.1) is calculated as $II(y_A,x_A) - II(y_B,x_B) - II(y_C,x_C) + II(y_D,x_D)$.

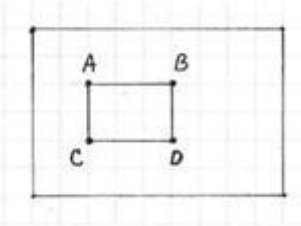


Fig.1 Image area integration using integral image

Detection happens inside a detection window. A minimum and maximum window size is chosen, and for each size a sliding step size is chosen. Then the detection window is moved across the image as follows:

1. Set the minimum window size, and sliding step corresponding to that size.
2. For the chosen window size, slide the window vertically and horizontally with the same step. At each step, a set of *N* face recognition filters is applied. If one filter gives a positive answer, the face is detected in the current widow.
3. If the size of the window is the maximum size stop the procedure. Otherwise increase the size of the window and corresponding sliding step to the next chosen size and go to the step 2.

Each face recognition filter (from the set of *N* filters) contains a set of cascade-connected classifiers. Each classifier looks at a rectangular subset of the detection window and determines if it looks like a face. If it does, the next classifier is applied. If all classifiers give a positive answer, the filter gives a positive answer and the face is recognized. Otherwise the next filter in the set of *N* filters is run.

Each classifier is composed of Haar feature extractors (weak classifiers). Each Haar feature is the weighted sum of 2-D integrals of small rectangular areas attached to each other. The weights may take values ±1. Fig.2 shows examples of Haar features relative to the enclosing detection window. Gray areas have a positive weight and white areas have a negative weight. Haar feature extractors are scaled with respect to the detection window size.

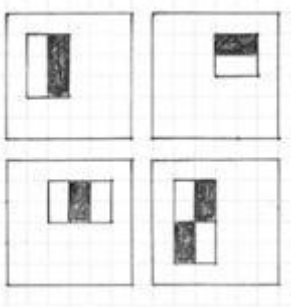


Fig.2 Example rectangle features shown relative to the enclosing detection window

The classifier decision is defined as:

$$C_m = \begin{cases} 1, & \sum_{i=0}^{I_m-1} F_{m,i} > \theta_m \\ 0, & \text{otherwise} \end{cases}$$

$$F_{m,i} = \begin{cases} \alpha_{m,i}, & \text{if } f_{m,i} > t_{m,i} \\ \beta_{m,i}, & \text{otherwise} \end{cases}$$

$f_{m,i}$ is the weighted sum of the 2-D integrals. is the decision threshold for the i-th feature extractor. $\alpha_{m,i}$ and $\beta_{m,i}$ are constant values associated with the i-th feature extractor. θ_m is the decision threshold for the m-th classifier.

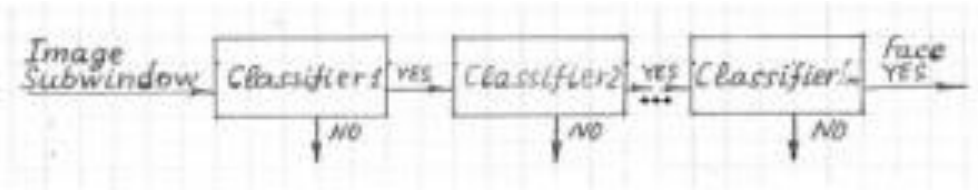


Fig.3 Object detection Viola-Jones filter

The cascade architecture is very efficient because the classifiers with the fewest features are placed at the beginning of the cascade, minimizing the total required computation. The most popular algorithm for features training is AdaBoost.

5. IMPLEMENTATION

The above proposed algorithm was implemented using Python using OpenCV. The application is ” pip ” installable and can be run through a single command from the terminal. The Detector module can be used in other programs to analyse video in real-time and obtain the eigen values for faces in the video which can be then used for other purposes.

5.1 Software Requirements

- Operating System = Windows
- Python version >= 3.4 is required.
- Python modules include OpenCV

5.2 Software Setup:

- **Install Python**
- **Install OpenCV**

OpenCV (Open Source Computer Vision) is a library aimed at building computer vision applications. It has numerous pre-written functions for image processing tasks. To install OpenCV, do a pip install of the library:

pip3 install opencv-python

- **Install face_recognition API;**

Finally, we will use face_recognition, dubbed as the world's simplest facial recognition API for Python. To install:

pip install dlib

pip install face_recognition

5.3 Working

Face Detection program using your Web Cam with Python.

We'd dealt with two files .

1. **haarcascade_frontalface_default.xml (** Basics of face detection using Haar Feature-based Cascade Classifiers**)**
2. **FaceDetection.py** (Python Script)

Object Detection using Haar feature-based cascade classifiers is an effective object detection method proposed by Paul Viola and Michael Jones in their paper, "Rapid Object Detection using a Boosted Cascade of Simple Features" in 2001.

In this paper, they have introduced the concept of Cascade of Classifiers. Haar-cascade Detection in OpenCV

Haar-cascade Detection in OpenCV

OpenCV comes with a trainer as well as a detector. If you want to train your own classifier for any object like a car, planes, etc. you can use OpenCV to create one. Here we will deal

with detection. OpenCV already contains many pre-trained classifiers for face, eyes, smile, etc. Those XML files are stored in opencv/data/haarcascades/ Folder.

5.3.1 Coding

First, we need to load the required XML classifiers.

```
import cv2
faceCascade = cv2.CascadeClassifier('haarcascade_frontalface_default.xml')
```

This loads the face cascade into memory so it's ready for use. Remember, the cascade is just an XML file that contains the data to detect faces.

```
video_capture = cv2.VideoCapture(0)
```

This line sets the video source to the default webcam, which OpenCV can easily capture.

```
while True:
    #Capture frame-by-frame
    ret, frame = video_capture.read()
```

In here, We're capturing the video. The read() method reads one frame from the video source which is the Webcam and it returns:

- The actual video frame read (one frame on each loop)
- A return code

If we run out of frames by any chance, The return code will tell us but in our scenario, We are not reading from a file it's our own Webcam so there's no possibility for us to run of out of frames.

```
gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
```

We just converted our Webcam feed to Grayscale. (Most of the operations in OpenCV are done in grayscale.)

Now

```
faces = faceCascade.detectMultiScale(
    gray,
    scaleFactor=1.5,
    minNeighbors=5,
    minSize=(30, 30),
    flags=cv2.CASCADE_SCALE_IMAGE
)
```

Here's when our code detects faces in our frame. Let's drill down from each of the above.

The *detectMultiScale* function is a general function that detects objects. It detects faces since we're using it on the face cascade.

The first option is the grayscale image.

The second is the *scaleFactor*. Since some faces may be closer to the camera, they would appear bigger than the faces in the back. The scale factor compensates for this. (You can change it to 1.5)

The detection algorithm uses a moving window to detect objects. *minNeighbors* defines how many objects are detected near the current one before it declares the face found. *minSize*, meanwhile, gives the size of each window.

The function returns a list of rectangles in which it believes it found a face. Next, we will loop over where it thinks it found something.

```
for (x, y, w, h) in faces:
    cv2.rectangle(frame, (x, y), (x+w, y+h), (0, 255, 0), 2)
```

This function returns 4 values: the x and y location of the rectangle, and the rectangle's width and height (w , h).

We use these values to draw a rectangle using the built-in rectangle() function.

```
cv2.imshow('FaceDetection', frame)
```

This is the frame name (Kind of similar to the title of the Webcam popup)

> Now, take a photo of my frame by Pressing the Space bar and exit the program by clicking 'ESC'.

```
#ESC Pressed
if k%256 == 27:
```

```
        break
#SPACE pressed
elif k%256 == 32:
        img_name = "facedetect_webcam_{}.png".format(img_counter)
        cv2.imwrite(img_name, frame)
        print("{} written!".format(img_name))
        img_counter += 1
```

img_counter is a variable to count and increment after each photo was taken. We should declare this before declaring our While loop on the beginning (While=True)

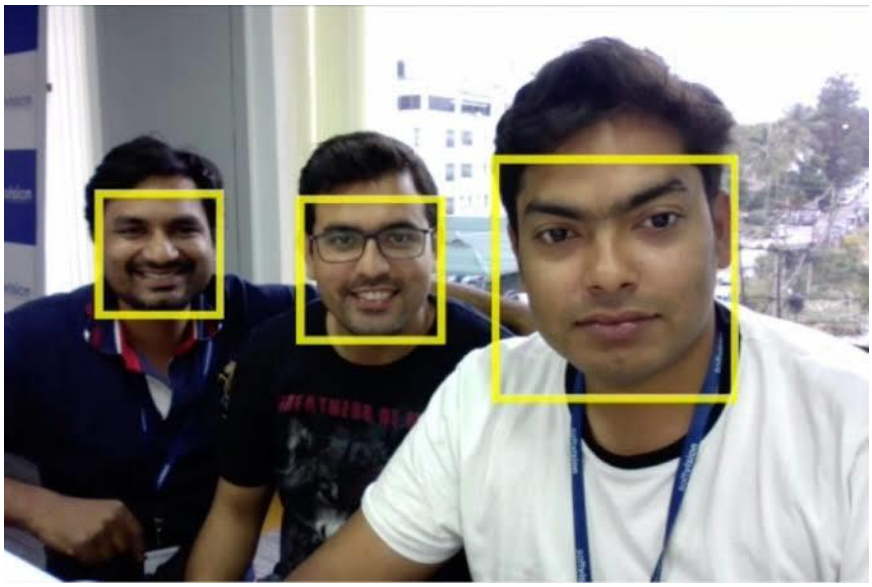
To Clean everything up;

```
# When everything is done, release the capture
video_capture.release()
cv2.destroyAllWindows()
```

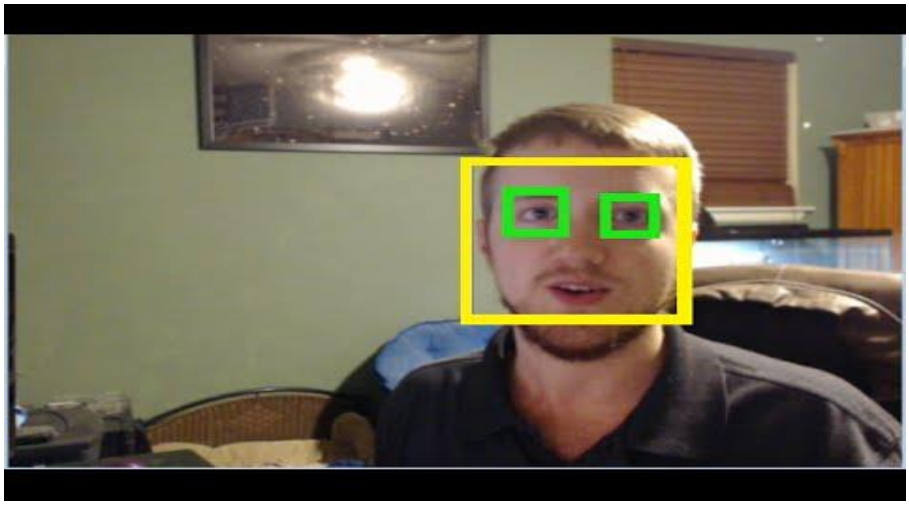
5.3.2 Snippets of the code :

```python
import cv2
import sys

faceCascade = cv2.CascadeClassifier('haarcascade_frontalface_default.xml')

video_capture = cv2.VideoCapture(0)

img_counter = 0

while True:
    # Capture frame-by-frame
    ret, frame = video_capture.read()

    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
    k = cv2.waitKey(1)
    faces = faceCascade.detectMultiScale(
        gray,
        scaleFactor=1.5,
        minNeighbors=5,
        minSize=(30, 30),
        flags=cv2.CASCADE_SCALE_IMAGE
    )

    # Draw a rectangle around the faces
    for (x, y, w, h) in faces:
        cv2.rectangle(frame, (x, y), (x+w, y+h), (0, 255, 0), 2)

    # Display the resulting frame
    cv2.imshow('FaceDetection', frame)

    if k%256 == 27: #ESC Pressed
        break
    elif k%256 == 32:
        # SPACE pressed
        img_name = "facedetect_webcam_{}.png".format(img_counter)
        cv2.imwrite(img_name, frame)
        print("{} written!".format(img_name))
        img_counter += 1


# When everything is done, release the capture
video_capture.release()
cv2.destroyAllWindows()
```

5.3.3. Simulation Result



No. of faces: 3



No. of faces: 1

6. CONCLUSION AND FULTURE SCOPE:

6.1 Conclusion

The approach presented here for face detection and tracking decreases the computation time producing results with high accuracy. Tracking of a face in a video sequence is done using KLT algorithm whereas Viola Jones is used for detecting facial features. Not only in video sequences, it has also been tested on live video using a webcam. Using this system many security and surveillance systems can be developed and required object can be traced down easily. In the coming days these algorithms can be used to detect a particular object rather than faces.

6.2 Future Scope

1. Future work is to work on the same domain but to track a particular face in a video sequence. That is like avoiding all other faces except the face required.

2. Viola Jones Algorithm is only meant for frontal and upright movement of the faces. It doesn't work when it comes to any arbitrary movement and hence, doesn't make sense. We would try to train classifiers so that it is subtle to all sort of movements.

3. We would definitely try to find the displacement of the Eigen vectors using Taylor Series.

7. REFERENCES

- https://opencv.org/
- https://github.com/nairitya03/Minor-Project
- https://www.pyimagesearch.com/
- https://docs.opencv.org/master/d9/df8/tutorial_root.html
- https://www.youtube.com/watch?v=kdLM6AOd2vc&list=PLS1QulWo1RIa7D1O6skqDQ-JZ1GGHKK-K
- https://youtu.be/-ZrDjwXZGxI