# Face Detection

Face detection is a computer vision problem that involves finding faces in photos. It is a trivial problem for humans to solve and has been solved reasonably well by classical feature-based techniques, such as the cascade classifier.

This is a C++ computer vision library that provides a python interface. The benefit of this implementation is that it provides pre-trained face detection models.

The pre trained models used in this task are:

1. haarcascade_frontalface_alt
2. haarcascade_frontalface_alt_tree
3. haarcascade_frontalface_alt2
4. haarcascade_frontalface_default

Below is the source code and some sample images which are tested with above listed models and the results are displayed below
Individually as per the model and finally based on the results the best model is given in the conclusion.

# Source Code

```
#importing opencv2 module
import cv2

#using pretrained models one by one, the cascade classifier takes file name as
argument
classifier =  cv2.CascadeClassifier(cv2.data.haarcascades +
'haarcascade_frontalface_alt.xml')

#reading a sample image
pic = cv2.imread('test9.jpg')

#resizing all images for consistent results and easy analysis
pic = cv2.resize(pic,(450,350))

#using the detectMultiScale function returns bounding boxes of detected faces
bboxes = classifier.detectMultiScale(pic)

#for each box a rectangle is been drawn to identify clearly
for box in bboxes:
   x,y,width,height = box
   x2,y2=x+width,y+height
   cv2.rectangle(pic,(x,y),(x2,y2),(255,255,0))



#To display the output picture with detected faces
cv2.imshow('facedetection',pic)

#wait key is used to tell how much time the output must be displayed
cv2.waitKey()
```

# Test Samples

Test1

Test2

Test3

Test4

Test5

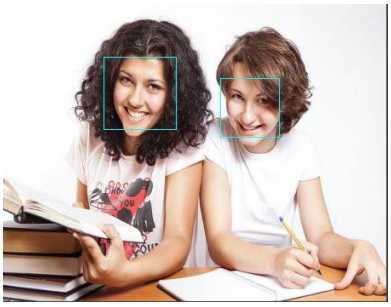
Test6

Test7

Test8

# haarcascade_frontalface_alt

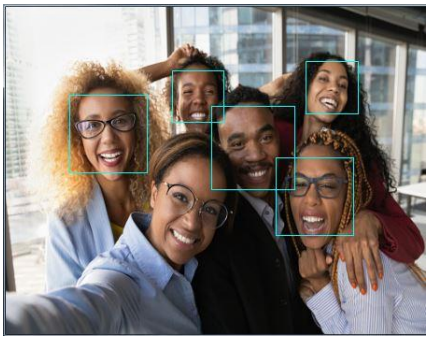
Test 1



Test2

Test3

Test4



Test5

Test6

Test7

Test8

# haarcascade_frontalface_alt_tree

Test1

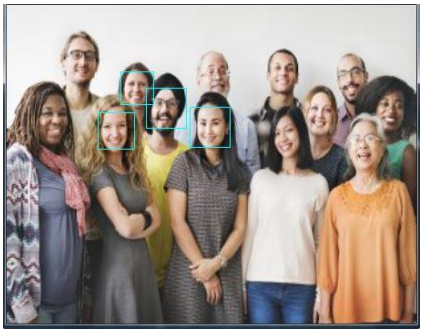
Test2

Test3

Test4

Test5


Test6



Test7

Test8

# haarcascade_frontalface_alt2

Test1

Test2

Test3

Test4

Test5

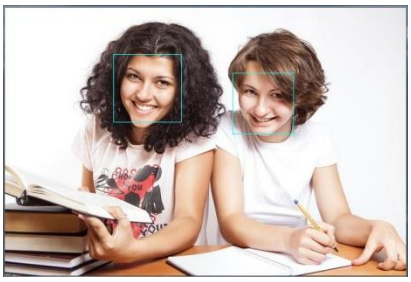

Test6

Test7

Test8

# haarcascade_frontalface_default

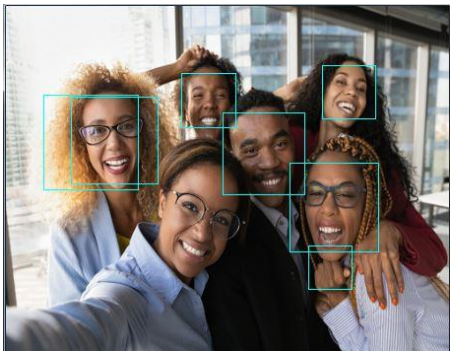
Test1



Test2

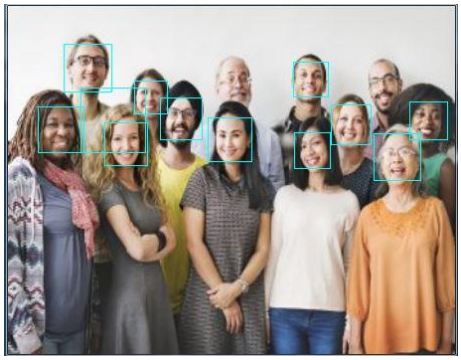
Test3

Test4



Test5


Test6



Test7

Test8

# Pros and Cons of each model

1. haarcascade_frontalface_alt
    - Pros
        - Detects faces in image with almost 70-90% accuracy
        - There is almost negligible false detection
    - Cons
        - All faces are not detected in a given image
2. haarcascade_frontalface_alt_tree
    - Pros
        - Detects faces in images when the images are taken at a long shot
    - Cons
        - Very less accurate in detecting images
3. haarcascade_frontalface_alt2
    - Pros
        - Detects faces in images when the images are taken at a long shot comparatively more accurate than haarcascade_frontalface_alt_tree
    - Cons
        - The accuracy is quite less in detecting faces
4. haarcascade_frontalface_default
    - Pros
        - Detects faces in images with high accuracy of about 70-90%
    - Cons
        - It has a little higher false detection as compared to haarcascade_frontalface_alt

# Conclusion

In my opinion haarcascade_frontalface_alt model would be the best, because as compared to other models it has higher accuracy in detecting images. Though haarcascade_frontalface_default also has almost similar accuracy, it has a higher false detection compared to haarcascade_frontalface_alt.