FaceDetectionFull-checkpoint

April 23, 2020

0.1 Import and Initialize

[1]:
```
import cv2
import matplotlib.pyplot as plt
import dlib
from imutils import face_utils
import numpy as np

font = cv2.FONT_HERSHEY_SIMPLEX

cascPath = "/home/vaibhav/PycharmProjects/untitled/venv/lib/python3.6/
 ↪site-packages/cv2/data/haarcascade_frontalface_default.xml"
eyePath = "/home/vaibhav/PycharmProjects/untitled/venv/lib/python3.6/
 ↪site-packages/cv2/data/haarcascade_eye.xml"
smilePath = "/home/vaibhav/PycharmProjects/untitled/venv/lib/python3.6/
 ↪site-packages/cv2/data/haarcascade_smile.xml"

faceCascade = cv2.CascadeClassifier(cascPath)
eyeCascade = cv2.CascadeClassifier(eyePath)
smileCascade = cv2.CascadeClassifier(smilePath)
```

0.2 1. Cascade Classifier

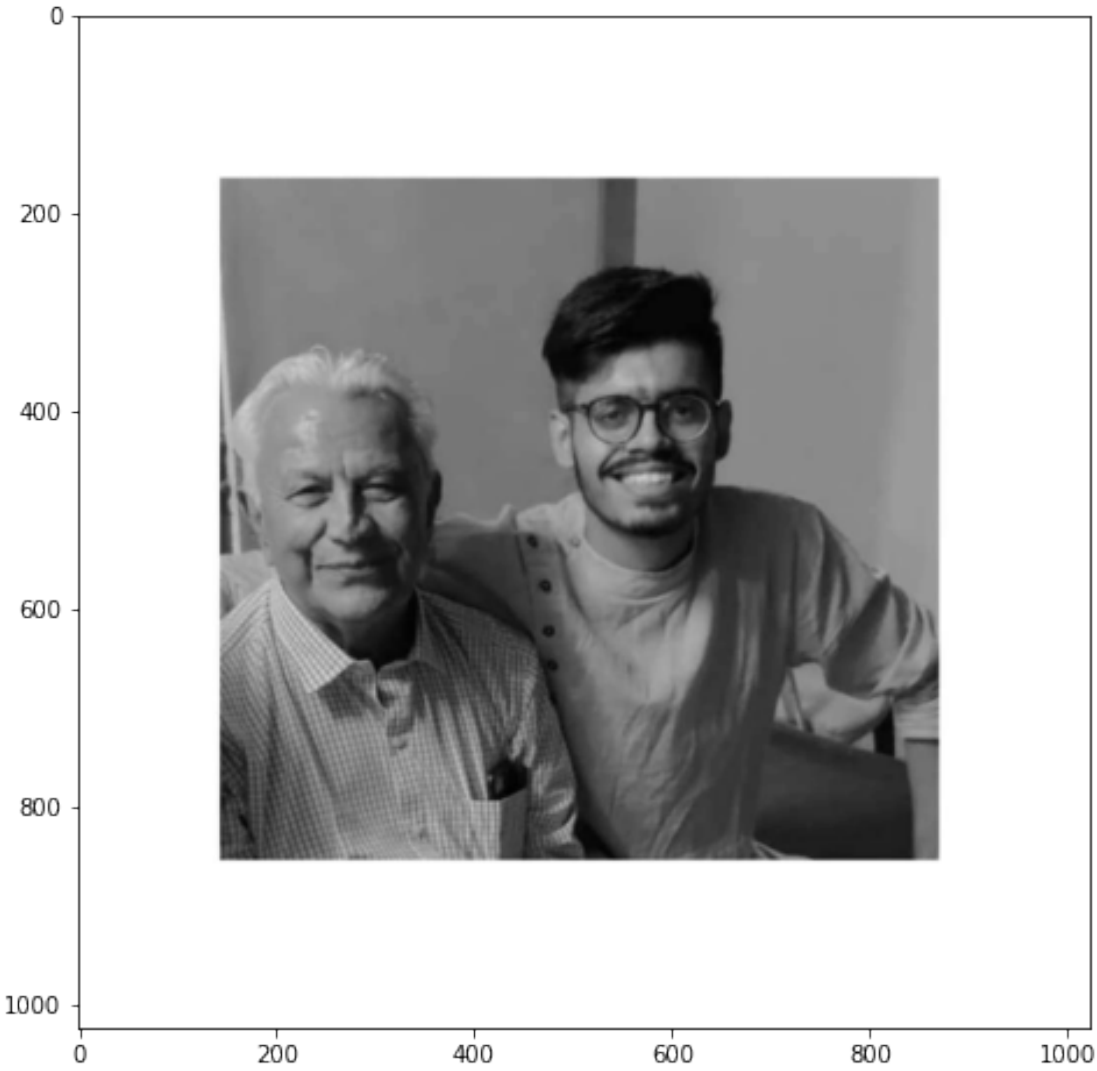
Cascade classifier, or namely cascade of boosted classifiers working with haar-like features, is a special case of ensemble learning, called boosting. It typically relies on Adaboost classifiers (and other models such as Real Adaboost, Gentle Adaboost or Logitboost).

Cascade classifiers are trained on a few hundred sample images of image that contain the object we want to detect, and other images that do not contain those images.

0.2.1 Detect face on an image

[2]:
```
# Load the image
gray = cv2.imread('test6.jpg', 0)

plt.figure(figsize=(12,8))
plt.imshow(gray, cmap='gray')
plt.show()
```

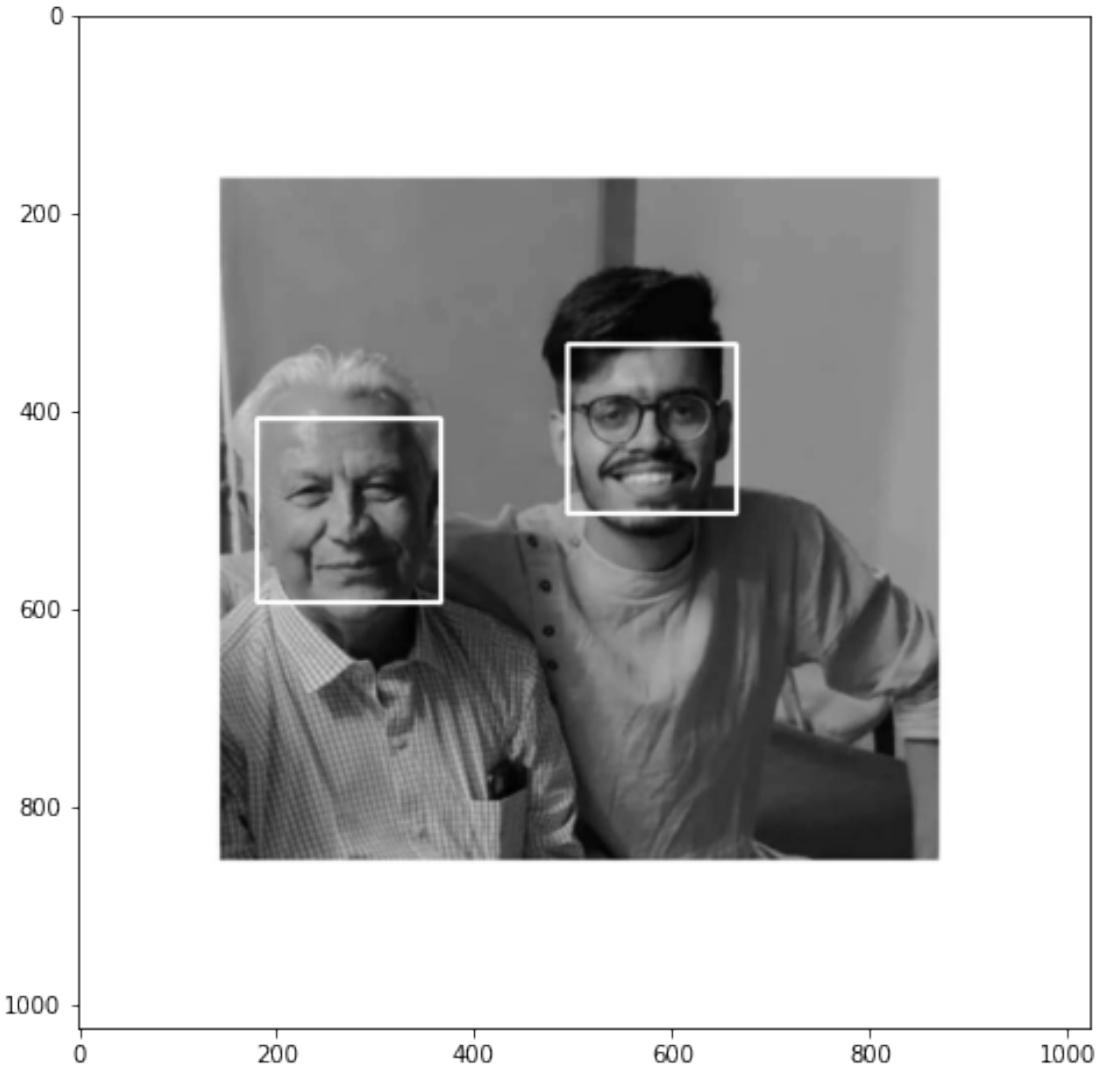


[3]:
```
# Detect faces
faces = faceCascade.detectMultiScale(
    gray,
    scaleFactor=1.1,
```

```
    minNeighbors=5,
    flags=cv2.CASCADE_SCALE_IMAGE
)

# For each face
for (x, y, w, h) in faces:
    # Draw rectangle around the face
    cv2.rectangle(gray, (x, y), (x+w, y+h), (255, 255, 255), 3)
```

[4]:
```
plt.figure(figsize=(12,8))
plt.imshow(gray, cmap='gray')
plt.show()
```



0.3 Real time face detection

[ ]:
```
# Launch Video Capture
video_capture = cv2.VideoCapture(0)

# While letter "q" not pressed
while True:

    # Capture video frame-by-frame
    ret, frame = video_capture.read()

    # Transform to gray scale
    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)

    # Detect faces
    faces = faceCascade.detectMultiScale(
        gray,
        scaleFactor=1.1,
        minNeighbors=5,
        #minSize=(30, 30),
        flags=cv2.CASCADE_SCALE_IMAGE
    )

    # For each face
    for (x, y, w, h) in faces:

        # Draw rectangle around the face
        cv2.rectangle(frame, (x, y), (x+w, y+h), (255, 0, 0), 3)
        cv2.putText(frame,'Face',(x, y), font, 2,(255,0,0),5)

        # Crop image to the face
        roi_gray = gray[y:y+h, x:x+w]
        roi_color = frame[y:y+h, x:x+w]

        # Detect the mouth
        smile = smileCascade.detectMultiScale(
            roi_gray,
            scaleFactor= 1.16,
            minNeighbors=40,
            minSize=(25, 25),
            flags=cv2.CASCADE_SCALE_IMAGE
            )
        # Put a rectangle and text around mouth
        for (sx, sy, sw, sh) in smile:
            cv2.rectangle(roi_color, (sh, sy), (sx+sw, sy+sh), (255, 0, 0), 2)
            cv2.putText(frame,'Smile',(x + sx,y + sy), 1, 1, (0, 255, 0), 1)
```

```
        # Detect the eyes
        eyes = eyeCascade.detectMultiScale(
            roi_gray,
            minSize=(10, 10),
            minNeighbors=20)

        # Put a rectangle and text around each exe
        for (ex,ey,ew,eh) in eyes:
            cv2.rectangle(roi_color,(ex,ey),(ex+ew,ey+eh),(0,255,0),2)
            cv2.putText(frame,'Eye',(x + ex,y + ey), 1, 1, (0, 255, 0), 1)

    # Count the number of faces
    cv2.putText(frame,'Number of Faces : ' + str(len(faces)),(40, 40), font,␣
↪1,(255,0,0),2)

    # Display the video output
    cv2.imshow('Video', frame)

    # Quit video by typing Q
    if cv2.waitKey(1) & 0xFF == ord('q'):
        break

# Release Capture
video_capture.release()
cv2.destroyAllWindows()
```

0.4 2. Dlib Histogram of Oriented Gradients

The histogram of oriented gradients (HOG) is a feature descriptor used in computer vision and image processing for the purpose of object detection. The technique counts occurrences of gradient orientation in localized portions of an image. This method is similar to that of edge orientation histograms, scale-invariant feature transform descriptors, and shape contexts, but differs in that it is computed on a dense grid of uniformly spaced cells and uses overlapping local contrast normalization for improved accuracy.

[5]:
```
thresh = 0.25
frame_check = 20
face_detect = dlib.get_frontal_face_detector()
```

HOG of the image

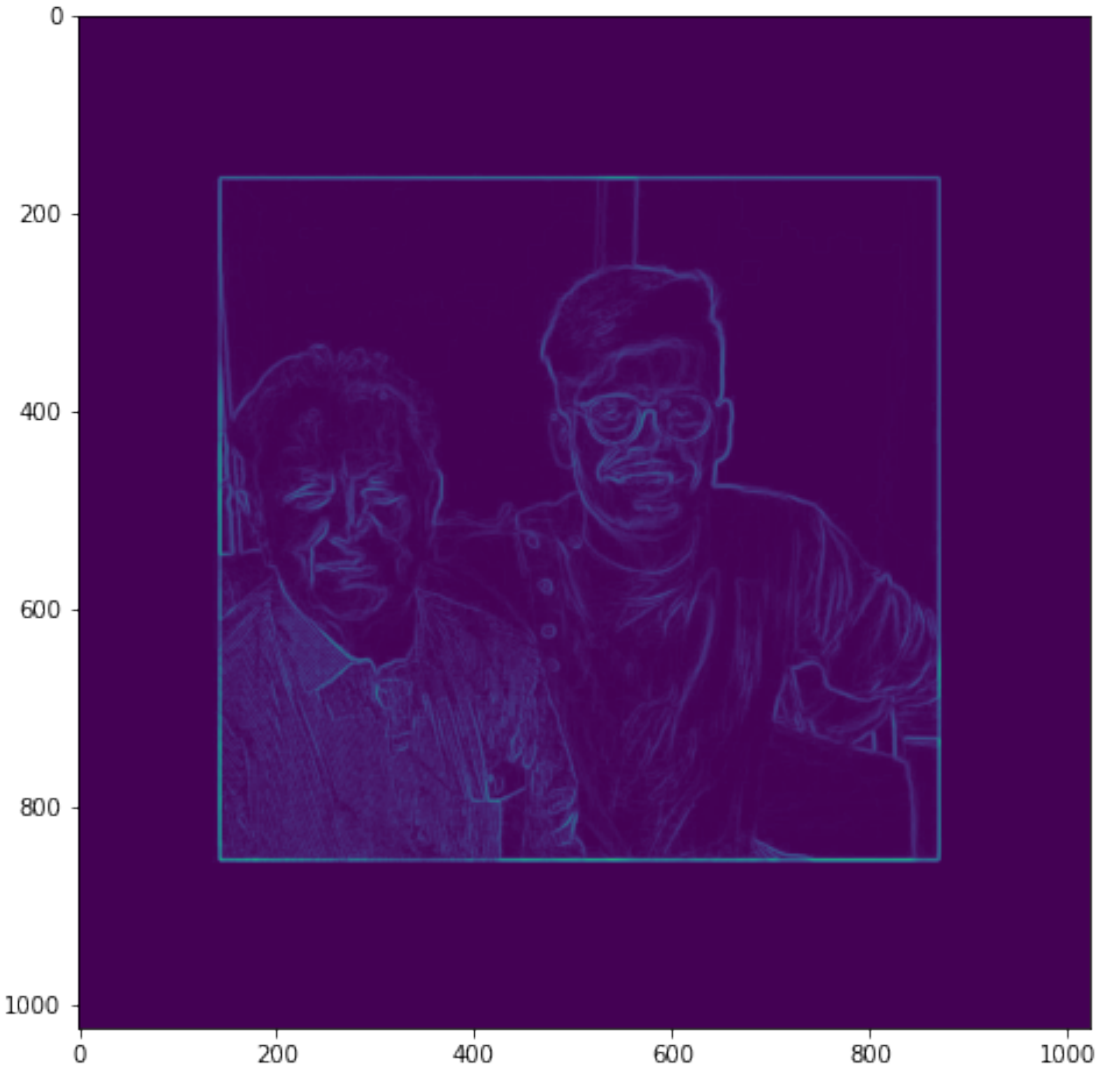
[7]:
```
gray = cv2.imread('test6.jpg', 0)

im = np.float32(gray) / 255.0

# Calculate gradient
```

```
gx = cv2.Sobel(im, cv2.CV_32F, 1, 0, ksize=1)
gy = cv2.Sobel(im, cv2.CV_32F, 0, 1, ksize=1)
mag, angle = cv2.cartToPolar(gx, gy, angleInDegrees=True)
```

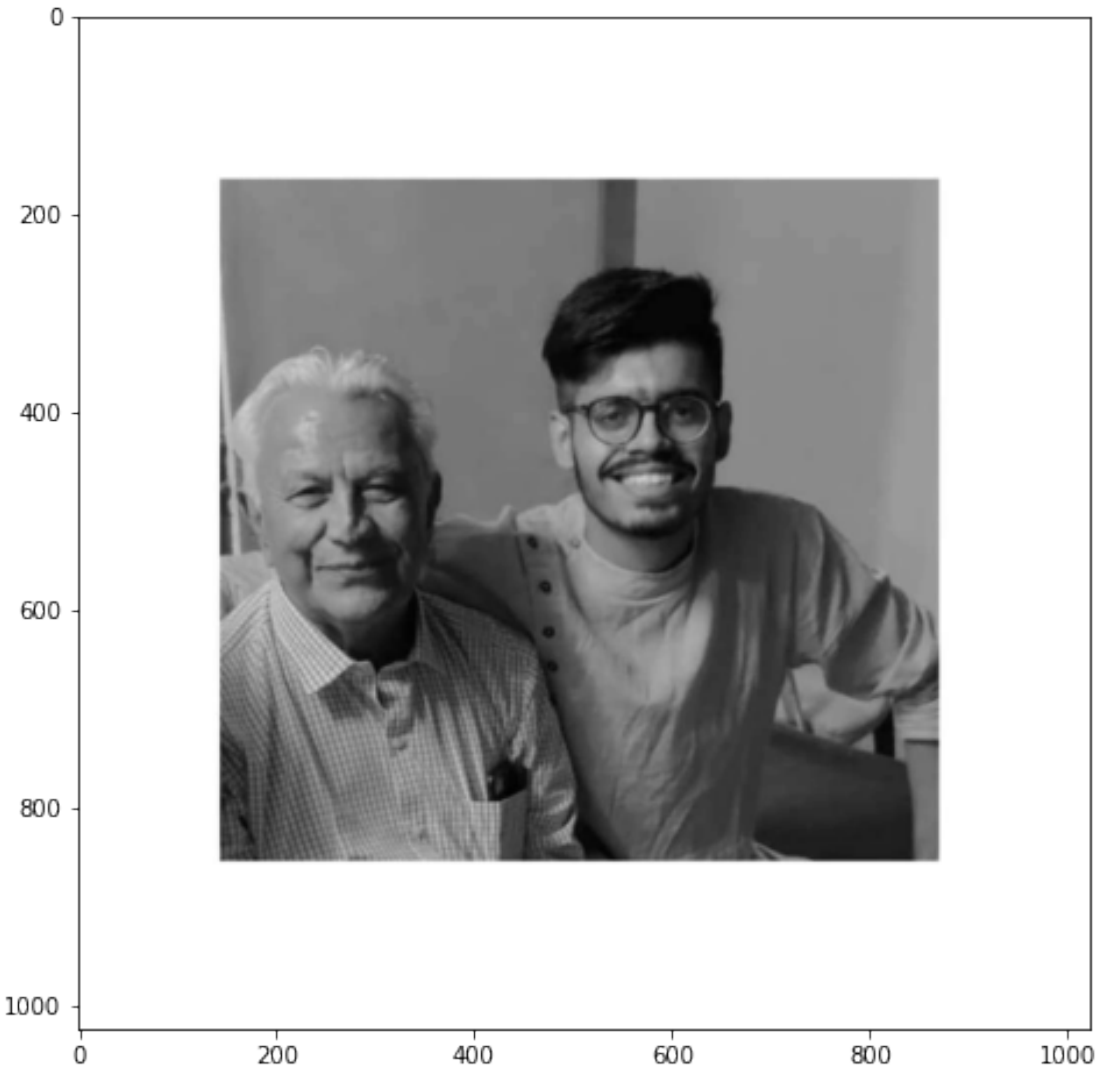
[8]:
```
plt.figure(figsize=(12,8))
plt.imshow(mag)
plt.show()
```



0.5 Detect face on an image

[10]:
```
# Load the image
gray = cv2.imread('test6.jpg', 0)

plt.figure(figsize=(12,8))
plt.imshow(gray, cmap='gray')
plt.show()
```

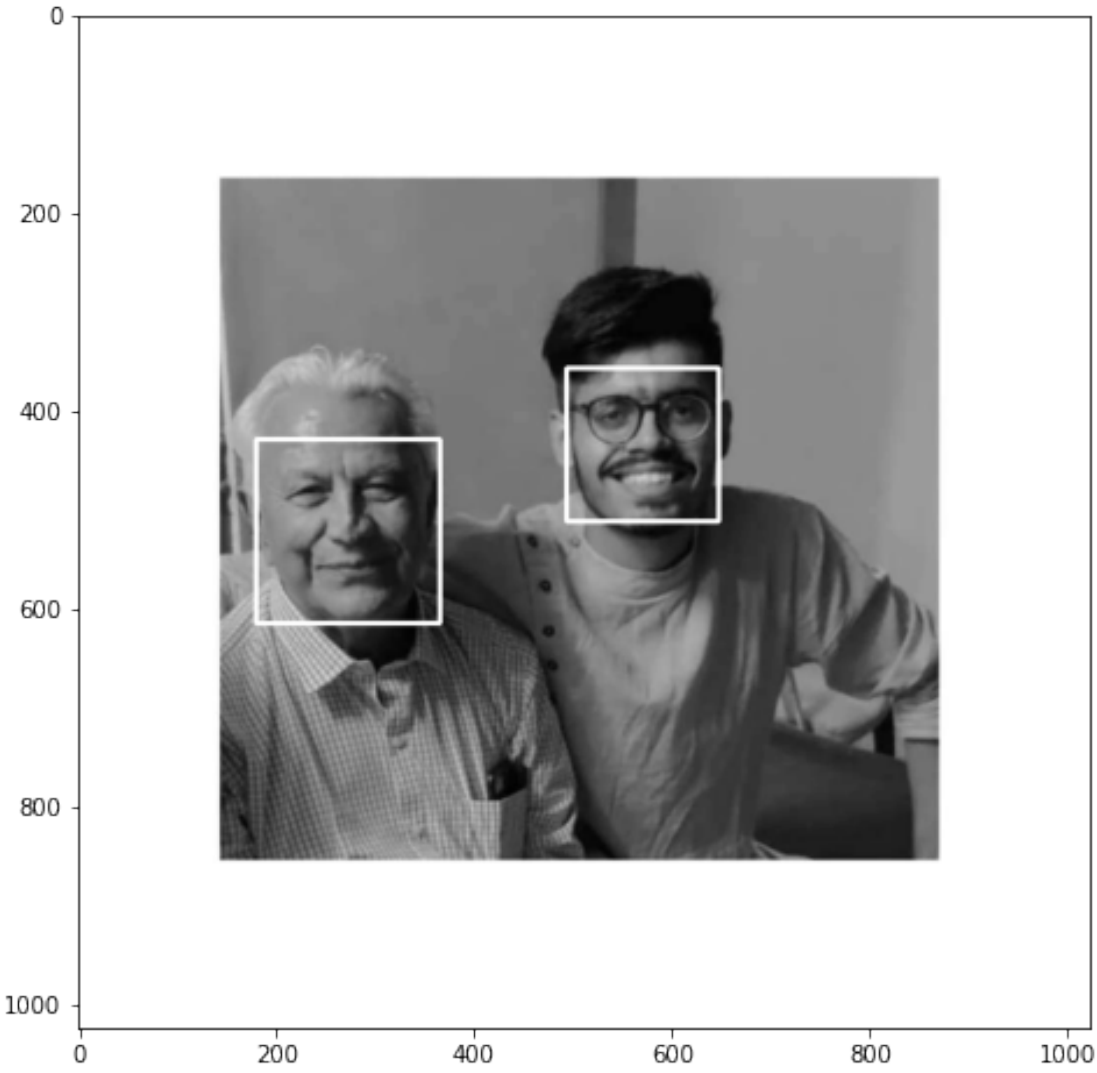


[11]:
```
rects = face_detect(gray, 1)

for (i, rect) in enumerate(rects):
        (x, y, w, h) = face_utils.rect_to_bb(rect)
```

```
        cv2.rectangle(gray, (x, y), (x + w, y + h), (255, 255, 255), 3)
```

[12]:
```
plt.figure(figsize=(12,8))
plt.imshow(gray, cmap='gray')
plt.show()
```



0.6 Real time face detection

[ ]:
```
video_capture = cv2.VideoCapture(0)
flag = 0

while True:
```

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

    face_index = 0

    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
    rects = face_detect(gray, 1)

    for (i, rect) in enumerate(rects):

        (x, y, w, h) = face_utils.rect_to_bb(rect)

        # Rectangle around the face
        cv2.rectangle(frame, (x, y), (x + w, y + h), (0, 255, 0), 2)

    # Display the video output
    cv2.imshow('Video', frame)

    # Quit video by typing Q
    if cv2.waitKey(1) & 0xFF == ord('q'):
        break

video_capture.release()
cv2.destroyAllWindows()
```

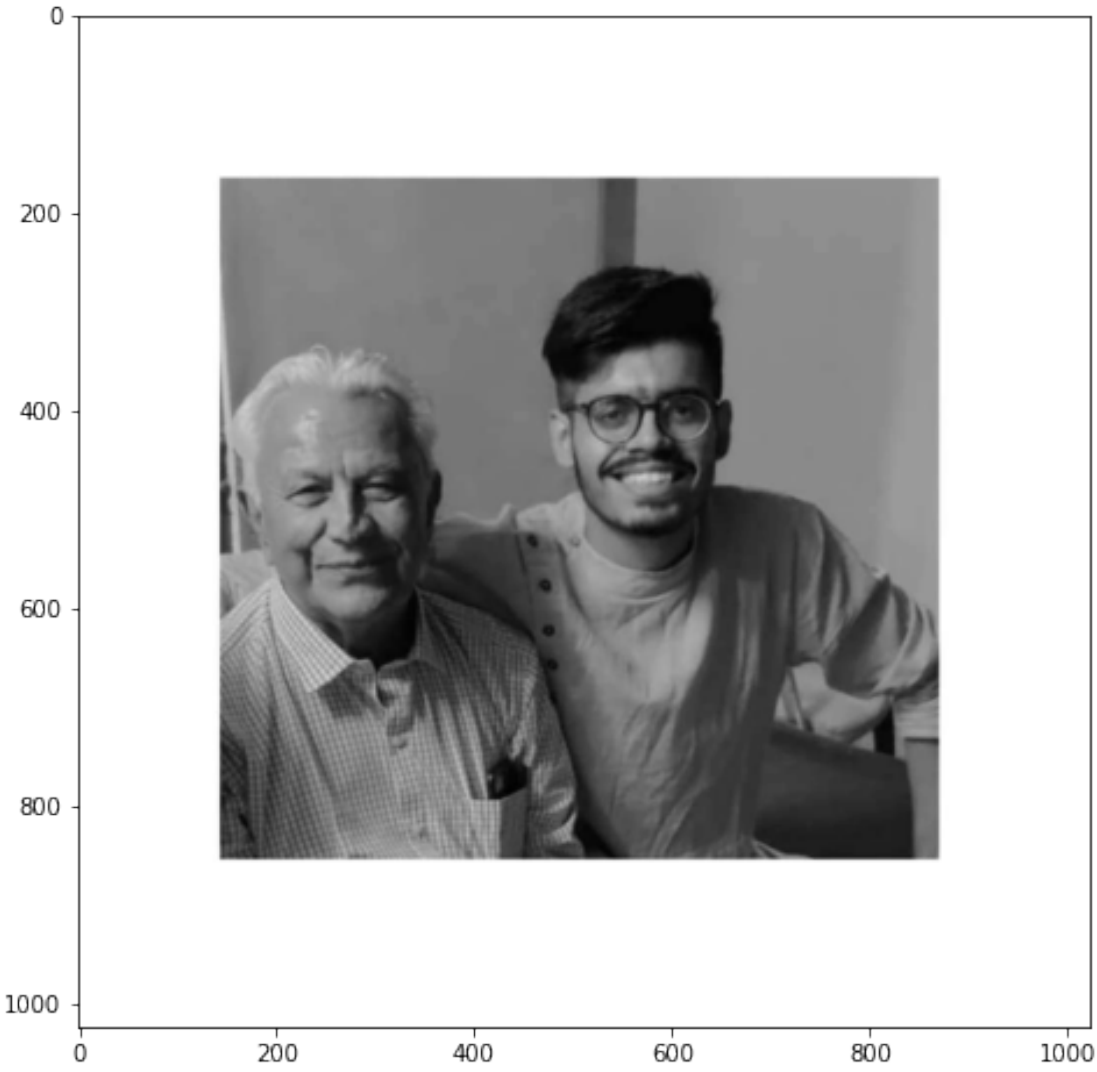
0.7 3. Deep Neural Network with DLib

This last method is based on Convolutional Neural Networks (CNN). It also implements a paper on Max-Margin Object Detection (MMOD) for enhanced results.

0.7.1 Detect a face on an image

[13]:
```
# Load the image
gray = cv2.imread('test6.jpg', 0)

plt.figure(figsize=(12,8))
plt.imshow(gray, cmap='gray')
plt.show()
```



[15]:
```
dnnFaceDetector = dlib.cnn_face_detection_model_v1("mmod_human_face_detector.
↪dat")
```

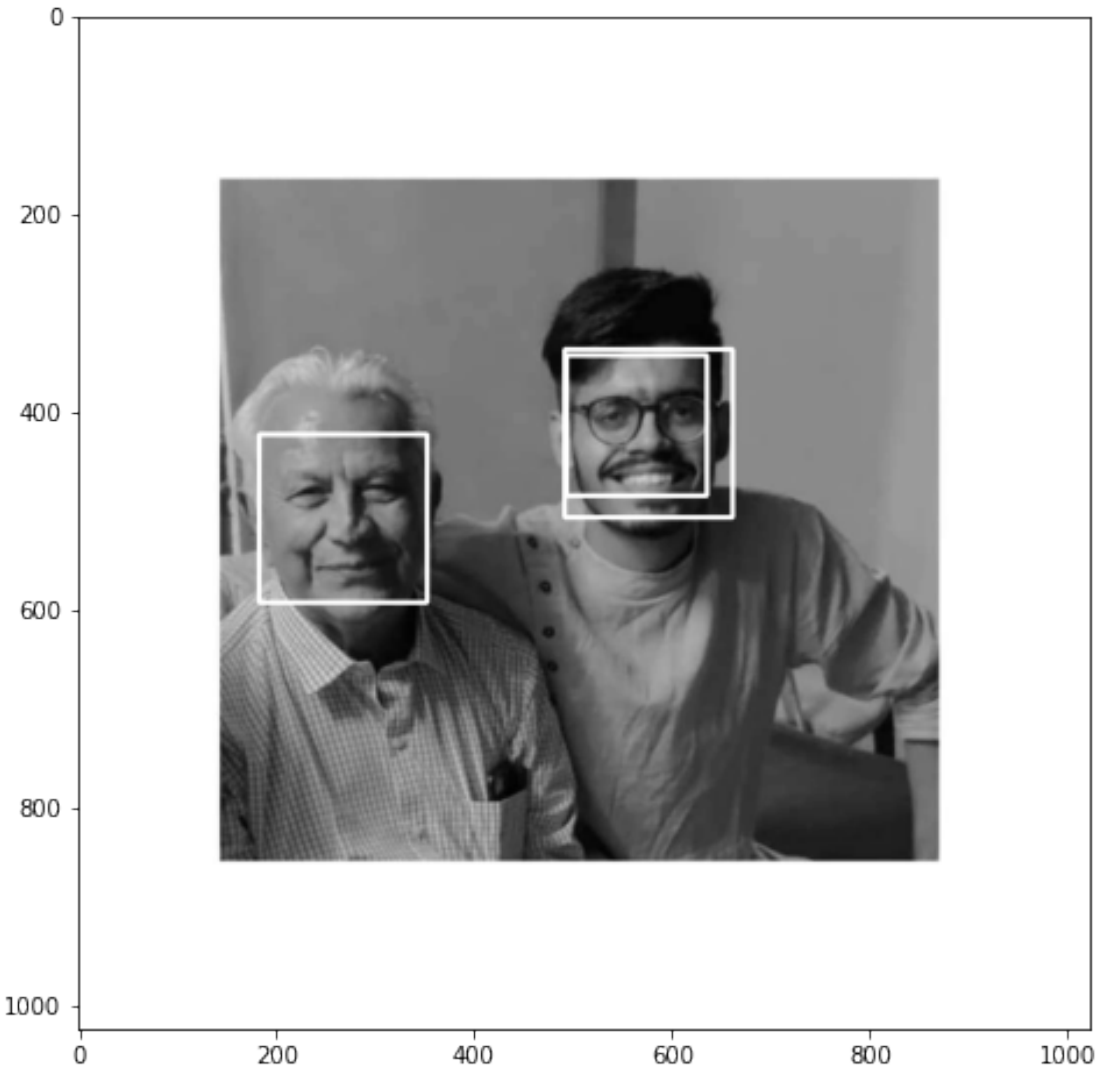
[17]:
```
rects = dnnFaceDetector(gray, 1)

for (i, rect) in enumerate(rects):

    x1 = rect.rect.left()
    y1 = rect.rect.top()
    x2 = rect.rect.right()
    y2 = rect.rect.bottom()

    # Rectangle around the face
    cv2.rectangle(gray, (x1, y1), (x2, y2), (255, 255, 255), 3)
```

[18]:
```
plt.figure(figsize=(12,8))
plt.imshow(gray, cmap='gray')
plt.show()
```



0.8 Real time face detection

[19]:
```
dnnFaceDetector = dlib.cnn_face_detection_model_v1("mmod_human_face_detector.
↪dat")
#faceRects = dnnFaceDetector(frameDlibHogSmall, 0)
```

[ ]:
```
video_capture = cv2.VideoCapture(0)
flag = 0
```

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

    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
    rects = dnnFaceDetector(gray, 1)

    for (i, rect) in enumerate(rects):

        x1 = rect.rect.left()
        y1 = rect.rect.top()
        x2 = rect.rect.right()
        y2 = rect.rect.bottom()

        # Rectangle around the face
        cv2.rectangle(frame, (x1, y1), (x2, y2), (0, 255, 0), 2)

    # Display the video output
    cv2.imshow('Video', frame)

    # Quit video by typing Q
    if cv2.waitKey(1) & 0xFF == ord('q'):
        break

video_capture.release()
cv2.destroyAllWindows()
```