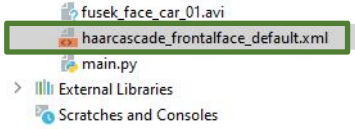
Face Detection - OpenCV




```python
def face_detect():
    cv2.namedWindow("face_detect", 0)
    video_cap = cv2.VideoCapture("fusek_face_car_01.avi")
    face_cascade = cv2.CascadeClassifier("haarcascade_frontalface_default.xml")

    while True:
        ret, frame = video_cap.read()
        paint_frame = frame.copy()
        if ret is True:
            faces = face_cascade.detectMultiScale(frame,
                                                  scaleFactor=1.2,
                                                  minNeighbors=3,
                                                  minSize=(100, 100),
                                                  maxSize=(500, 500))
            for one_face in faces:
                cv2.rectangle(paint_frame, one_face, (0, 0, 255), 12)
                cv2.rectangle(paint_frame, one_face, (255, 255, 255), 4)

        cv2.imshow("opencv_frame", paint_frame)
        if cv2.waitKey(2) == ord("q"):
            break
```

Face Detection - OpenCV

Python:

cv.CascadeClassifier.detectMultiScale(image[, scaleFactor[, minNeighbors[, flags[, minSize[, maxSize]]]]]) -> objects

cv.CascadeClassifier.detectMultiScale2(image[, scaleFactor[, minNeighbors[, flags[, minSize[, maxSize]]]]]) -> objects, numDetections

cv.CascadeClassifier.detectMultiScale3(image[, scaleFactor[, minNeighbors[, flags[, minSize[, maxSize[, outputRejectLevels]]]]]]) -> objects, rejectLevels, levelWeights

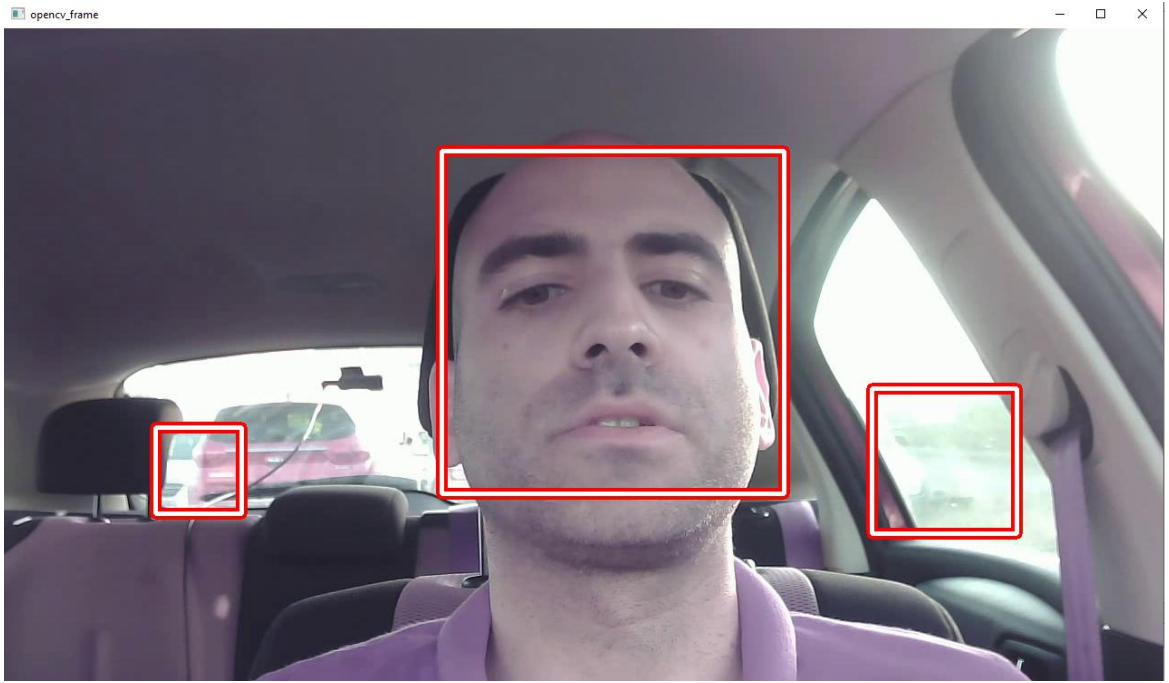
Detects objects of different sizes in the input image. The detected objects are returned as a list of rectangles.

Parameters

image	Matrix of the type CV_8U containing an image where objects are detected.
objects	Vector of rectangles where each rectangle contains the detected object, the rectangles may be partially outside the original image.
scaleFactor	Parameter specifying how much the image size is reduced at each image scale.
minNeighbors	Parameter specifying how many neighbors each candidate rectangle should have to retain it.
flags	Parameter with the same meaning for an old cascade as in the function cvHaarDetectObjects. It is not used for a new cascade.
minSize	Minimum possible object size. Objects smaller than that are ignored.
maxSize	Maximum possible object size. Objects larger than that are ignored. If `maxSize == minSize` model is evaluated on single scale.

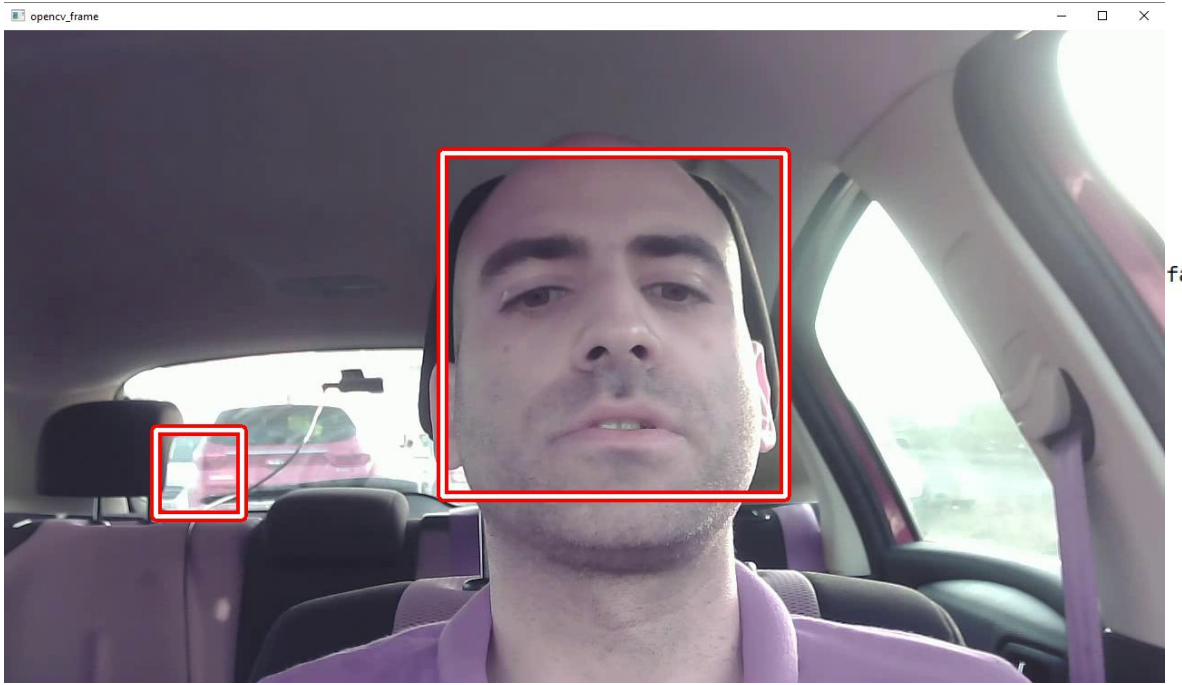
https://docs.opencv.org/4.9.0/d1/de5/classcv_1_1CascadeClassifier.html

Face Detection - OpenCV



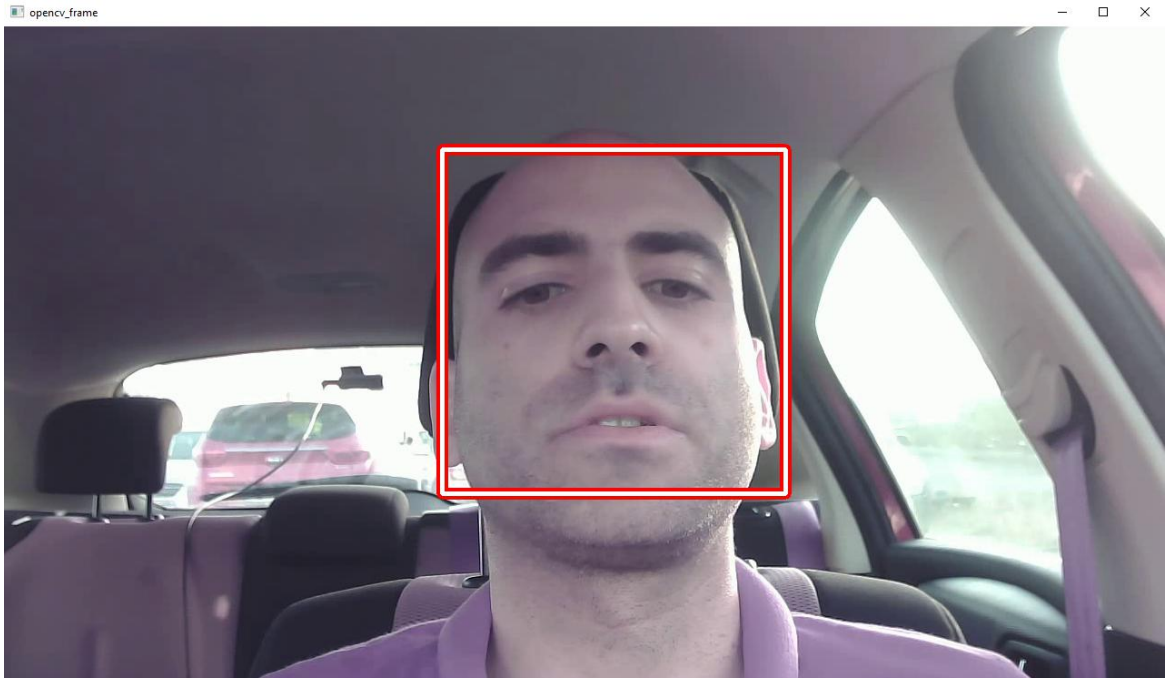

```
faces = face_cascade.detectMultiScale(frame,
                                      scaleFactor=1.1,
                                      minNeighbors=1,
                                      minSize=(50, 50),
                                      maxSize=(500, 500))
```

Face Detection - OpenCV




```
faces = face_cascade.detectMultiScale(frame,
                                      scaleFactor=1.1,
                                      minNeighbors=3,
                                      minSize=(50, 50),
                                      maxSize=(500, 500))
```

Face Detection - OpenCV




```
faces = face_cascade.detectMultiScale(frame,
                                      scaleFactor=1.1,
                                      minNeighbors=3,
                                      minSize=(100, 100),
                                      maxSize=(500, 500))
```

Face Detection - TASK

TASK

1. Create the combination of various (at least two) face cascade classifiers
   - profile_face + frontal_face
   - classifiers: http://mrl.cs.vsb.cz/data/vyuka/zao/haarcascades.zip or https://github.com/kipr/opencv/tree/master/data/haarcascades
   - experiment with different type of **detectMultiScale** function – use **levelWeights** to filter weaker classifications
   - measurement of the time required for localization

```
cv.CascadeClassifier.detectMultiScale(  image[, scaleFactor[, minNeighbors[, flags[, minSize[, maxSize]]]]] -> objects
)
cv.CascadeClassifier.detectMultiScale2( image[, scaleFactor[, minNeighbors[, flags[, minSize[, maxSize]]]]] -> objects, numDetections
)
cv.CascadeClassifier.detectMultiScale3( image[, scaleFactor[, minNeighbors[, flags[, minSize[, maxSize[, outputRejectLevels]]]]]] -> objects, rejectLevels, levelWeights
```

2. Detection of eye and mouth/smile inside each face region
   - you can use the following eye detector: http://mrl.cs.vsb.cz/data/vyuka/zao/eye_cascade_fusek.zip

3. You can use the following video or create new: http://mrl.cs.vsb.cz/data/vyuka/zao/fusek_face_car_01.zip

4. *Bonus* - Try recognition of close/open eyes (detection of sclera or pupil/iris)
   - for example, using thresholding: https://docs.opencv.org/4.9.0/da/d97/tutorial_threshold_inRange.html
   - or Template Matching, or Hough Circle Transform: https://docs.opencv.org/4.9.0/d4/d70/tutorial_hough_circle.html

CascadeClassifier docs: https://docs.opencv.org/4.9.0/d1/de5/classcv_1_1CascadeClassifier.html