we're going to discuss object detection with Haar Cascades. We'll do face and eye detection to start. In order to do object recognition/detection with cascade files, you first need cascade files. For the extremely popular tasks, these already exist. Detecting things like faces, cars, smiles, eyes, and license plates for example are all pretty prevalent.

First, I will show you how to use these cascade files, then I will show you how to embark on creating your very own cascades, so that you can detect any object you want, which is pretty darn cool!

You can use Google to find various Haar Cascades of things you may want to detect. You shouldn't have too much trouble finding the aforementioned types. We will use a [Face cascade](https://github.com/Itseez/opencv/blob/master/data/haarcascades/haarcascade_frontalface_default.xml) and [Eye cascade](https://github.com/Itseez/opencv/blob/master/data/haarcascades/haarcascade_eye.xml). You can find a few more at the root directory of Haar cascades. Note the license for using/distributing these Haar Cascades.

Let's begin our code. I am assuming you have downloaded the haarcascade\_eye.xml and haarcascade\_frontalface\_default.xml from the links above, and have these files in your project's directory.

import numpy as np

import cv2

# multiple cascades: https://github.com/Itseez/opencv/tree/master/data/haarcascades

#https://github.com/Itseez/opencv/blob/master/data/haarcascades/haarcascade\_frontalface\_default.xml

face\_cascade = cv2.CascadeClassifier('haarcascade\_frontalface\_default.xml')

#https://github.com/Itseez/opencv/blob/master/data/haarcascades/haarcascade\_eye.xml

eye\_cascade = cv2.CascadeClassifier('haarcascade\_eye.xml')

cap = cv2.VideoCapture(0)

Here, we begin with import cv2 and numpy, then we load in our face and eye cascades. Simple enough so far.

while 1:

ret, img = cap.read()

gray = cv2.cvtColor(img, cv2.COLOR\_BGR2GRAY)

faces = face\_cascade.detectMultiScale(gray, 1.3, 5)

Now we begin our typical loop, the only new thing here is the creation of faces. For more information, visit the [documentation for the detectMultiScale functionality](https://docs.opencv.org/2.4/modules/objdetect/doc/cascade_classification.html#cascadeclassifier-detectmultiscale). Basically, it finds faces! We also want to find eyes, but, in a world of false positives, wouldn't it be prudent to logically make it so that we only find eyes in faces? Let's hope we're not looking for eyes that aren't in faces! In all seriousness, "eye detection" probably wouldn't find an eyeball laying around. Most eye detection uses the surrounding skin, eye lids, eye lashes, and eye brows to also make the detection. Thus, our next step is to break down the faces first, before getting to the eyes:

for (x,y,w,h) in faces:

cv2.rectangle(img,(x,y),(x+w,y+h),(255,0,0),2)

roi\_gray = gray[y:y+h, x:x+w]

roi\_color = img[y:y+h, x:x+w]

Here, we're finding faces, their sizes, drawing rectangles, and noting the ROI. Next, we poke around for some eyes:

eyes = eye\_cascade.detectMultiScale(roi\_gray)

for (ex,ey,ew,eh) in eyes:

cv2.rectangle(roi\_color,(ex,ey),(ex+ew,ey+eh),(0,255,0),2)

If we find those, we'll go ahead and make some more rectangles. Next we finish up:

cv2.imshow('img',img)

k = cv2.waitKey(30) & 0xff

if k == 27:

break

cap.release()

cv2.destroyAllWindows()

Full code:

import numpy as np

import cv2

# multiple cascades: https://github.com/Itseez/opencv/tree/master/data/haarcascades

#https://github.com/Itseez/opencv/blob/master/data/haarcascades/haarcascade\_frontalface\_default.xml

face\_cascade = cv2.CascadeClassifier('haarcascade\_frontalface\_default.xml')

#https://github.com/Itseez/opencv/blob/master/data/haarcascades/haarcascade\_eye.xml

eye\_cascade = cv2.CascadeClassifier('haarcascade\_eye.xml')

cap = cv2.VideoCapture(0)

while 1:

ret, img = cap.read()

gray = cv2.cvtColor(img, cv2.COLOR\_BGR2GRAY)

faces = face\_cascade.detectMultiScale(gray, 1.3, 5)

for (x,y,w,h) in faces:

cv2.rectangle(img,(x,y),(x+w,y+h),(255,0,0),2)

roi\_gray = gray[y:y+h, x:x+w]

roi\_color = img[y:y+h, x:x+w]

eyes = eye\_cascade.detectMultiScale(roi\_gray)

for (ex,ey,ew,eh) in eyes:

cv2.rectangle(roi\_color,(ex,ey),(ex+ew,ey+eh),(0,255,0),2)

cv2.imshow('img',img)

k = cv2.waitKey(30) & 0xff

if k == 27:

break

cap.release()

cv2.destroyAllWindows()

Not bad. You may notice I had to remove my glasses. These caused some trouble. My mouth also was detected pretty often as eyes, sometimes even a face, but you get the idea. Facial hair and other things can often fool rudimentary face detection, and even advanced. Skin color can also cause a lot of trouble, as we often try to simplify images as much as possible, thus losing a lot of color values. There's even a mini-industry around avoiding face detection and recognition. The website [CVDazzle](https://cvdazzle.com/" \t "blank) would be an example of this. Some of these are pretty outlandish, but they can work pretty well. You could also always go the route of complete facial reconstructive surgery to avoid automated tracking and detection, so there's always that, but that's a bit more permanent. A hair-do is a bit more temporary and easy to do.