Image Processing with Computer Vision

Name: Prashanth B

Fundamentals

```
#Reading images and video
import cv2
from google.colab.patches import cv2_imshow
```

```
img = cv2.imread('/content/cat.jpg')
cv2_imshow(img)
```

```
img.size
```

```
819840
```

```
img.shape
```

```
(427, 640, 3)
```

```
img.dtype
```

```
dtype('uint8')
```

```
from matplotlib import pyplot as plt            # this lets you draw inline pictures in the notebooks
import pylab                                    # this allows you to control figure size
pylab.rcParams['figure.figsize'] = (10.0, 8.0)  # this controls figure size in the notebook
```

```
plt.imshow(img)
```

```
<matplotlib.figure.AxesImage at 0x7ff10a260ac0>
```

```
cv2.split(img)
```

```
(array([[0, 0, 0, ..., 0, 0, 0],
        [0, 0, 0, ..., 0, 0, 0],
        [0, 0, 0, ..., 0, 0, 0],
        ...,
        [0, 0, 0, ..., 0, 0, 0],
        [0, 0, 0, ..., 0, 0, 0],
        [0, 0, 0, ..., 0, 0, 0]], dtype=uint8),
 array([[0, 0, 0, ..., 0, 0, 0],
        [0, 0, 0, ..., 0, 0, 0],
        [0, 0, 0, ..., 0, 0, 0],
        ...,
        [0, 0, 0, ..., 0, 0, 0],
        [0, 0, 0, ..., 0, 0, 0],
        [0, 0, 0, ..., 0, 0, 0]], dtype=uint8),
 array([[0, 0, 0, ..., 0, 0, 0],
        [0, 0, 0, ..., 0, 0, 0],
        [0, 0, 0, ..., 0, 0, 0],
        ...,
        [0, 0, 0, ..., 0, 0, 0],
        [0, 0, 0, ..., 0, 0, 0],
        [0, 0, 0, ..., 0, 0, 0]], dtype=uint8))
```

```
len(cv2.split(img))
```

```
3
```

```
#blue channel
b = cv2.split(img)[0]
b
```

```
array([[0, 0, 0, ..., 0, 0, 0],
       [0, 0, 0, ..., 0, 0, 0],
       [0, 0, 0, ..., 0, 0, 0],
       ...,
       [0, 0, 0, ..., 0, 0, 0],
       [0, 0, 0, ..., 0, 0, 0],
       [0, 0, 0, ..., 0, 0, 0]], dtype=uint8)
```

```
b.shape
```

```
(427, 640)
```

```
b.dtype
```

```
dtype('uint8')
```

```
#green channel
g = cv2.split(img)[1]
g
```

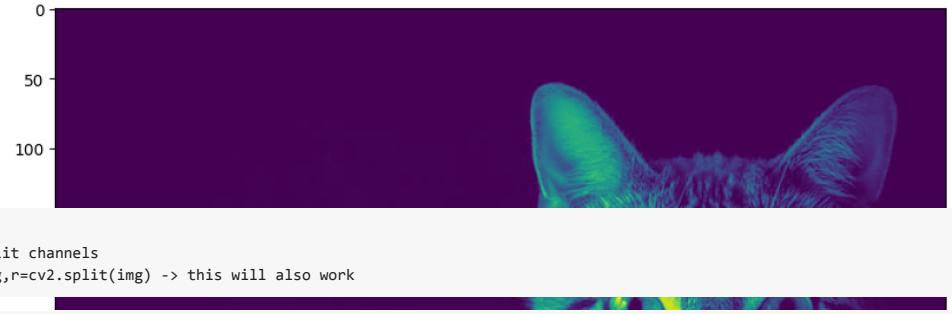
```
array([[0, 0, 0, ..., 0, 0, 0],
       [0, 0, 0, ..., 0, 0, 0],
       [0, 0, 0, ..., 0, 0, 0],
       ...,
       [0, 0, 0, ..., 0, 0, 0],
       [0, 0, 0, ..., 0, 0, 0],
       [0, 0, 0, ..., 0, 0, 0]], dtype=uint8)
```

```
#red channel
r = cv2.split(img)[2]
r
```

```
array([[0, 0, 0, ..., 0, 0, 0],
       [0, 0, 0, ..., 0, 0, 0],
       [0, 0, 0, ..., 0, 0, 0],
       ...,
       [0, 0, 0, ..., 0, 0, 0],
       [0, 0, 0, ..., 0, 0, 0],
       [0, 0, 0, ..., 0, 0, 0]], dtype=uint8)
```

```
plt.imshow(g)
```

```
<matplotlib.image.AxesImage at 0x7ff0f9379180>
```

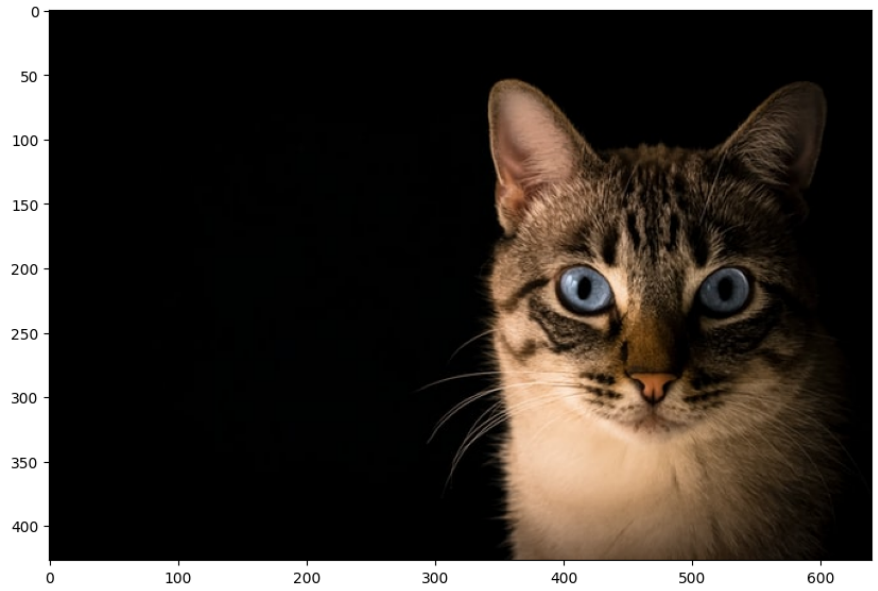



```
#or
# split channels
# b,g,r=cv2.split(img) -> this will also work
```



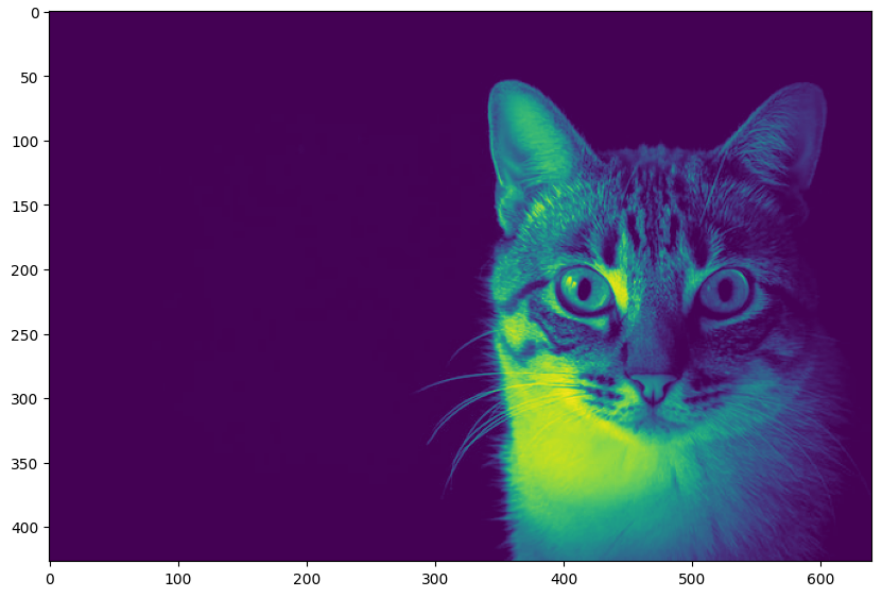
```
merged=cv2.merge([r,g,b])
# merge takes an array of single channel matrices
plt.imshow(merged)
```

```
<matplotlib.image.AxesImage at 0x7ff0f95d71f0>
```




```
opencv_merged=cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
plt.imshow(opencv_merged)
```

```
<matplotlib.image.AxesImage at 0x7ff0f91633d0>
```




```
img.shape
```

```
(427, 640, 3)
```

```
img
```

```
array([[[0, 0, 0],
        [0, 0, 0],
        [0, 0, 0],
        ...,
        [0, 0, 0],
        [0, 0, 0],
        [0, 0, 0]],

       [[0, 0, 0],
        [0, 0, 0],
        [0, 0, 0],
        ...,
        [0, 0, 0],
        [0, 0, 0],
        [0, 0, 0]],

       [[0, 0, 0],
        [0, 0, 0],
        [0, 0, 0],
        ...,
        [0, 0, 0],
        [0, 0, 0],
        [0, 0, 0]],

       ...,

       [[0, 0, 0],
        [0, 0, 0],
        [0, 0, 0],
        ...,
        [0, 0, 0],
        [0, 0, 0],
        [0, 0, 0]],

       [[0, 0, 0],
        [0, 0, 0],
        [0, 0, 0],
        ...,
        [0, 0, 0],
        [0, 0, 0],
        [0, 0, 0]],

       [[0, 0, 0],
        [0, 0, 0],
        [0, 0, 0],
        ...,
        [0, 0, 0],
        [0, 0, 0],
        [0, 0, 0]]], dtype=uint8)
```

Image Processing

```
vertical_flip_img = cv2.flip(img, 0)
plt.imshow(vertical_flip_img)
```

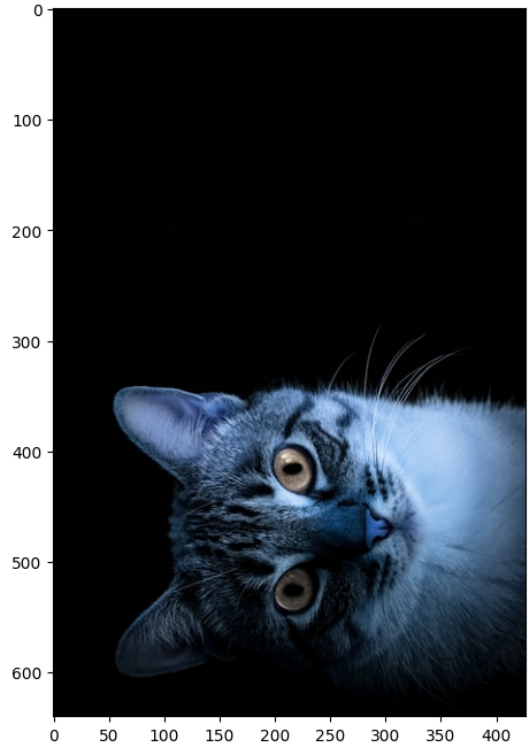
```
<matplotlib.image.AxesImage at 0x7ff0f8e36bf0>
```


```
horizontal_flip_img = cv2.flip(img, 1)
plt.imshow(horizontal_flip_img)
```

```
<matplotlib.image.AxesImage at 0x7ff0f90cf4f0>
```


```
transposed_img=cv2.transpose(img)
plt.imshow(transposed_img)
```

```
<matplotlib.image.AxesImage at 0x7ff0f8d80b20>
```




```
'''Minimum, maximum
To find the min or max of a matrix, you can use minMaxLoc.
This takes a single channel image (it doesn't make much sense to take the max of a 3 channel image).
So in the next code snippet you see a for loop, using python style image slicing,
to look at each channel of the input image separately.'''

for i in range(3):
   min_value, max_value, min_location, max_location=cv2.minMaxLoc(img[:,:,i])
   print("min {} is at {}, and max {} is at {}".format(min_value, min_location, max_value, max_location))
```

```
  min 0.0 is at (0, 0), and max 255.0 is at (405, 208)
  min 0.0 is at (0, 0), and max 238.0 is at (405, 208)
  min 0.0 is at (0, 0), and max 255.0 is at (437, 208)
```

```
img[:,:,1]
```

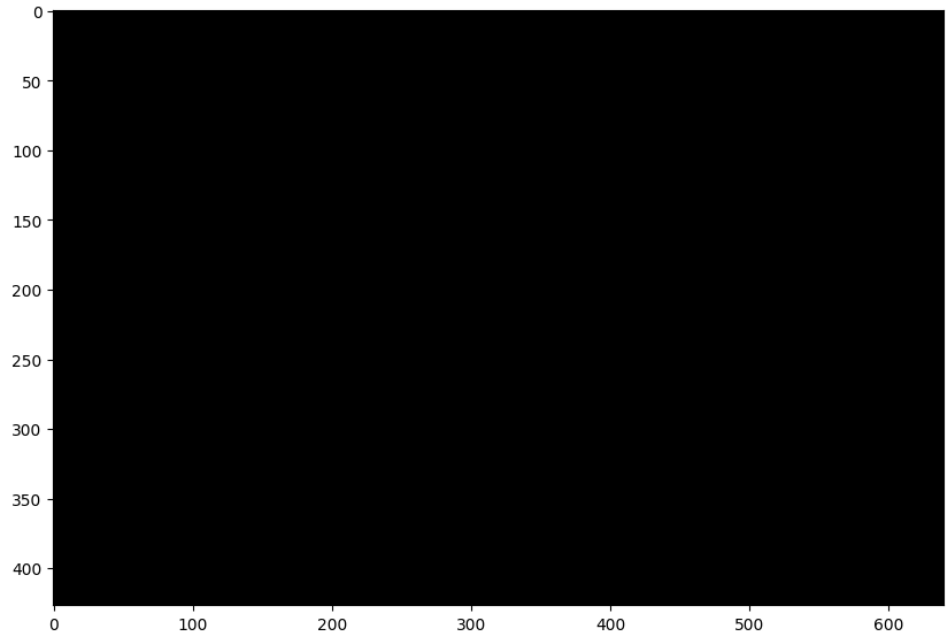
```
array([[0, 0, 0, ..., 0, 0, 0],
       [0, 0, 0, ..., 0, 0, 0],
       [0, 0, 0, ..., 0, 0, 0],
       ...,
       [0, 0, 0, ..., 0, 0, 0],
       [0, 0, 0, ..., 0, 0, 0],
       [0, 0, 0, ..., 0, 0, 0]], dtype=uint8)
```

```
#Arithmetic operations on images

import numpy as np
blank_image = np.zeros((shape:=img.shape), dtype:=np.uint8)
blank_image.shape
```

```
(427, 640, 3)
```

```
plt.imshow(blank_image)
```

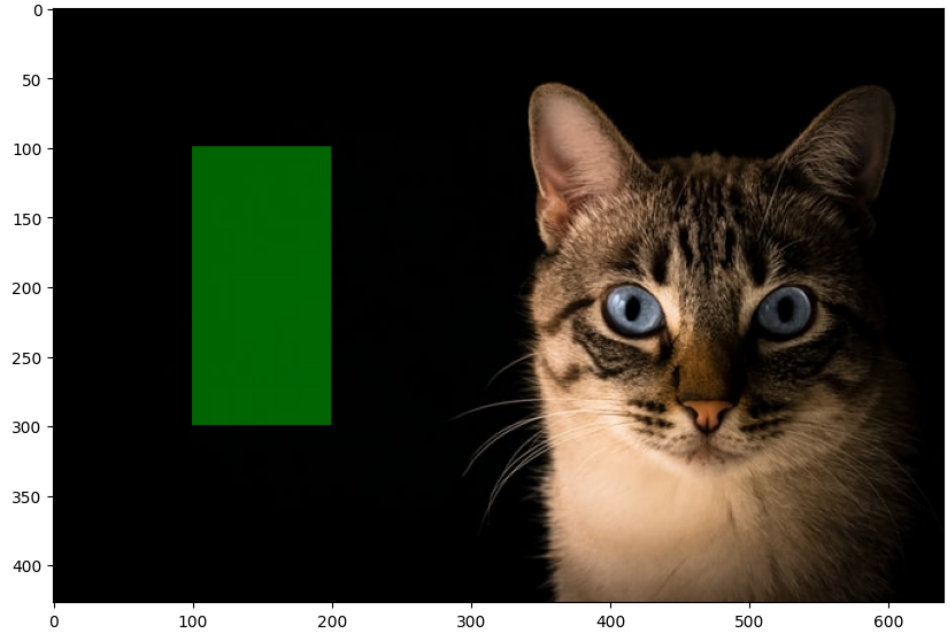
```
<matplotlib.image.AxesImage at 0x7ff0f8c5d0f0>
```




```
blank_image[100:300,100:200,1]=100; #give it a green square
```

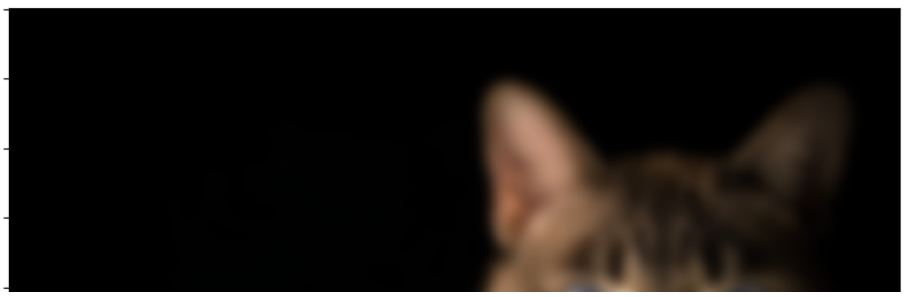
```
new_image=cv2.add(blank_image,img) # add the two images together
plt.imshow(cv2.cvtColor(new_image, cv2.COLOR_BGR2RGB))
```

```
<matplotlib.image.AxesImage at 0x7ff0f86bce20>
```




```
d=15
img_blur3 = cv2.GaussianBlur(src=img, ksize=(2*d+1, 2*d+1), sigmaX=-1)
plt.imshow(cv2.cvtColor(img_blur3, cv2.COLOR_BGR2RGB))
```

```
<matplotlib.image.AxesImage at 0x7ff0f85fb880>
```

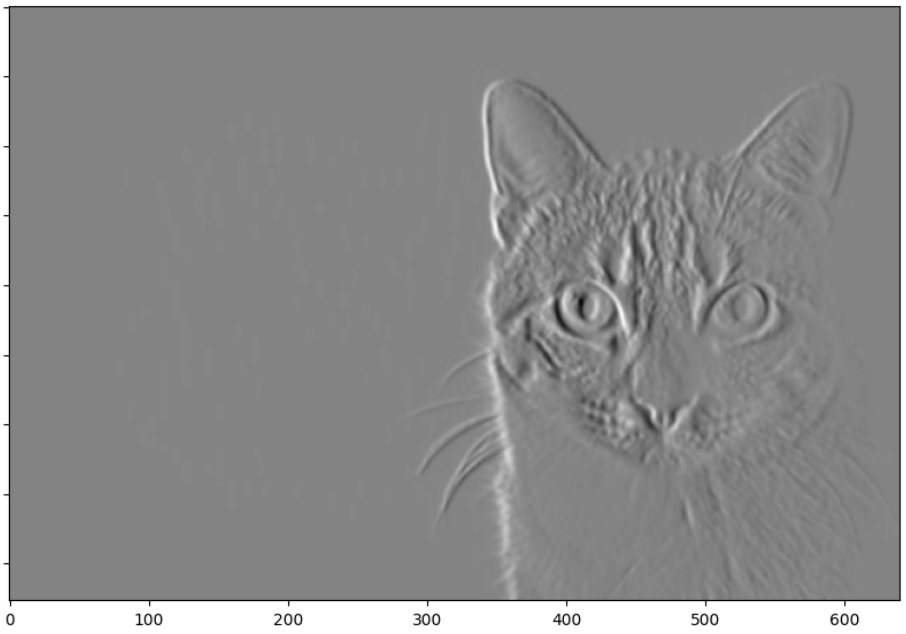


Edge Detection

```
sobelimage=cv2.cvtColor(img,cv2.COLOR_BGR2GRAY)
sobelx = cv2.Sobel(sobelimage,cv2.CV_64F,1,0,ksize=9)
sobely = cv2.Sobel(sobelimage,cv2.CV_64F,0,1,ksize=9)
```

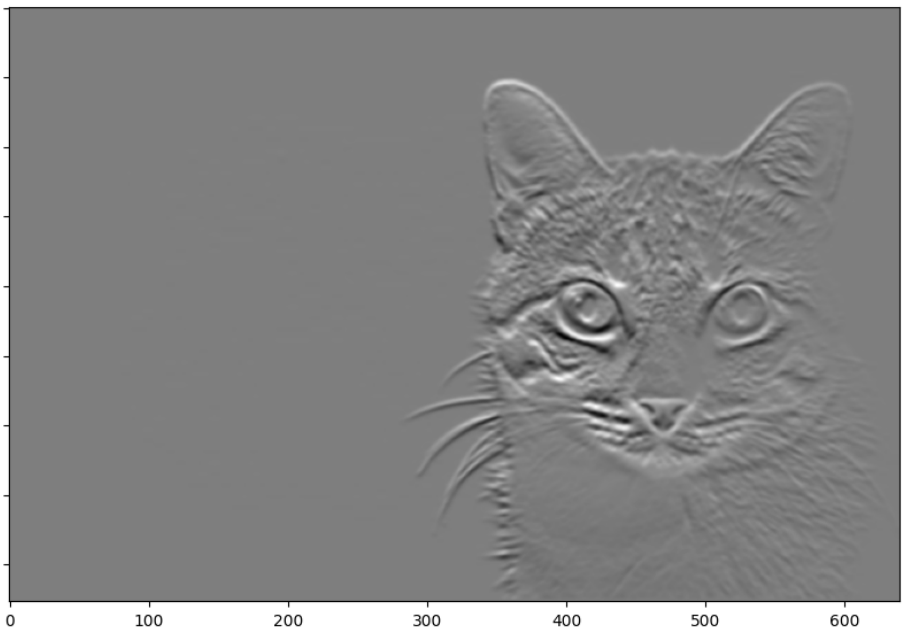
```
plt.imshow(sobelx, cmap='gray')
```

```
<matplotlib.image.AxesImage at 0x7ff0f873a500>
```



```
plt.imshow(sobely, cmap='gray')
```

```
<matplotlib.image.AxesImage at 0x7ff0f84b6080>
```



```
#Canny Edge Detection

th1=30
th2=90  #Canny recommends threshold2 is (3 times) threshold1 - you could try experimenting with this...
```

```
d=3     #gaussian blur

edge_result=img.copy()
edge_result = cv2.GaussianBlur(edge_result, (2*d+1, 2*d+1), -1)
gray = cv2.cvtColor(edge_result, cv2.COLOR_BGR2GRAY)
edge = cv2.Canny(gray, th1, th2)

edge_result[edge != 0] = (0, 0, 255)
#this takes pixels in edge_result where edge non-zero
#colours them to bright red

plt.imshow(cv2.cvtColor(edge_result, cv2.COLOR_BGR2RGB))
```

```
<matplotlib.image.AxesImage at 0x7ff0f81bd270>
```

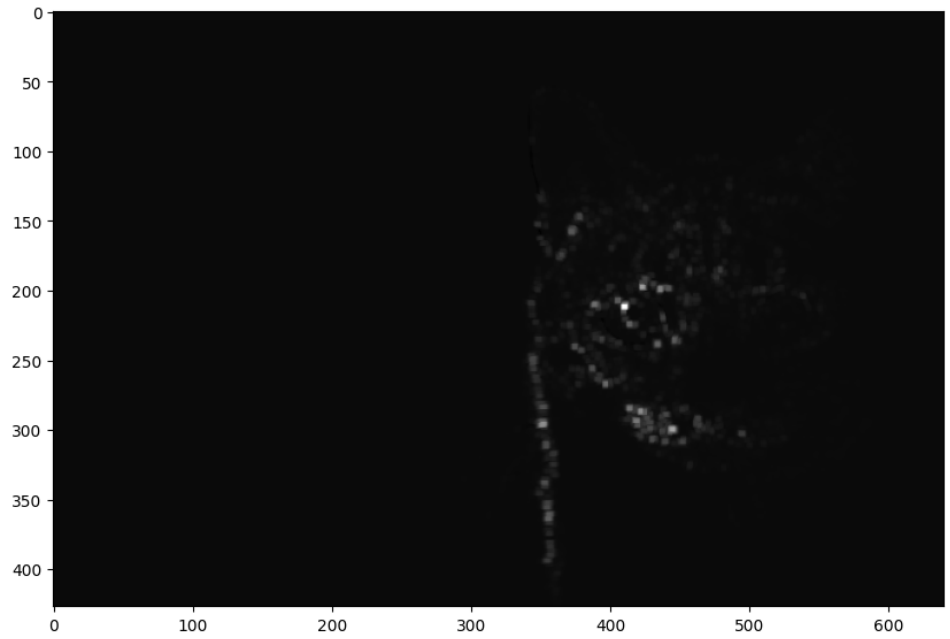
Features in Computer Vision

```
harris_test=img.copy()
gray = cv2.cvtColor(harris_test,cv2.COLOR_BGR2GRAY)   #greyscale it
gray = np.float32(gray)

blocksize=4
kernel_size=5           #sobel kernel: must be odd and fairly small
thr = 0.01

dst = cv2.cornerHarris(gray, blocksize, kernel_size, thr)
# parameters are blocksize, Sobel parameter and Harris threshold

dst = cv2.dilate(dst, None)
#result is dilated for marking the corners, this is visualisation related and just makes them bigger

harris_test[dst>0.01*dst.max()]=[0,0,255]
plt.imshow(cv2.cvtColor(harris_test, cv2.COLOR_BGR2RGB))
#we then plot these on the input image for visualisation purposes, using bright red
```

```
<matplotlib.image.AxesImage at 0x7ff0f790f5b0>
```

```
plt.imshow(dst, cmap='gray')
```

```
<matplotlib.image.AxesImage at 0x7ff0f82990c0>
```



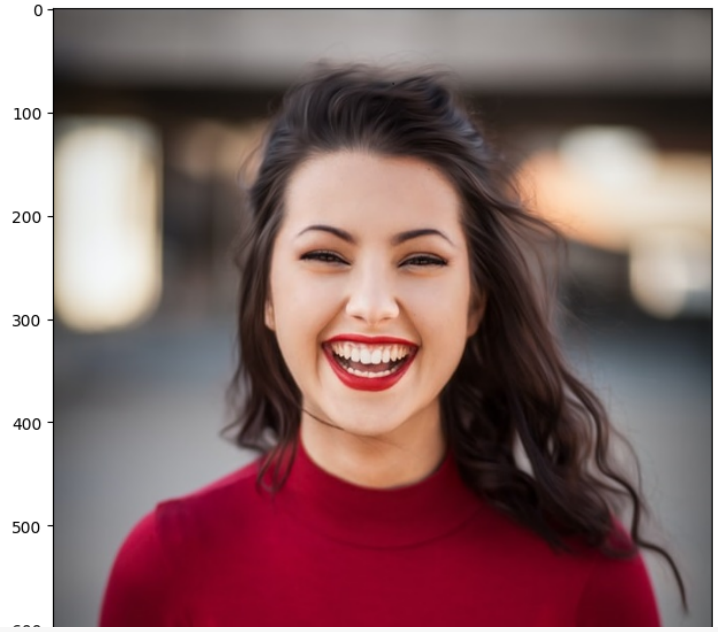
Cascade Classification

```
#download util files

!wget --no-check-certificate \
    https://raw.githubusercontent.com/computationalcore/introduction-to-opencv/master/assets/haarcascade_frontalface_default.xml \
    -O haarcascade_frontalface_default.xml
!wget --no-check-certificate \
    https://raw.githubusercontent.com/computationalcore/introduction-to-opencv/master/assets/haarcascade_smile.xml \
    -O haarcascade_smile.xml
!wget --no-check-certificate \
    https://raw.githubusercontent.com/computationalcore/introduction-to-opencv/master/assets/haarcascade_eye.xml \
    -O haarcascade_eye.xml
```

```
--2023-06-28 08:54:26--  https://raw.githubusercontent.com/computationalcore/introduction-to-opencv/master/assets/haarcascade_frontalface_default.xml
Resolving raw.githubusercontent.com (raw.githubusercontent.com)... 185.199.108.133, 185.199.110.133, 185.199.111.133, ...
Connecting to raw.githubusercontent.com (raw.githubusercontent.com)|185.199.108.133|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 930127 (908K) [text/plain]
Saving to: ‘haarcascade_frontalface_default.xml’

haarcascade_frontal 100%[===================>] 908.33K  --.-KB/s    in 0.06s

2023-06-28 08:54:27 (14.2 MB/s) - ‘haarcascade_frontalface_default.xml’ saved [930127/930127]

--2023-06-28 08:54:27--  https://raw.githubusercontent.com/computationalcore/introduction-to-opencv/master/assets/haarcascade_smile.xml
Resolving raw.githubusercontent.com (raw.githubusercontent.com)... 185.199.110.133, 185.199.108.133, 185.199.109.133, ...
Connecting to raw.githubusercontent.com (raw.githubusercontent.com)|185.199.110.133|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 188506 (184K) [text/plain]
Saving to: ‘haarcascade_smile.xml’

haarcascade_smile.x 100%[===================>] 184.09K  --.-KB/s    in 0.03s

2023-06-28 08:54:27 (6.21 MB/s) - ‘haarcascade_smile.xml’ saved [188506/188506]

--2023-06-28 08:54:27--  https://raw.githubusercontent.com/computationalcore/introduction-to-opencv/master/assets/haarcascade_eye.xml
Resolving raw.githubusercontent.com (raw.githubusercontent.com)... 185.199.108.133, 185.199.111.133, 185.199.109.133, ...
Connecting to raw.githubusercontent.com (raw.githubusercontent.com)|185.199.108.133|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 341406 (333K) [text/plain]
Saving to: ‘haarcascade_eye.xml’

haarcascade_eye.xml 100%[===================>] 333.40K  --.-KB/s    in 0.04s

2023-06-28 08:54:27 (8.29 MB/s) - ‘haarcascade_eye.xml’ saved [341406/341406]
```

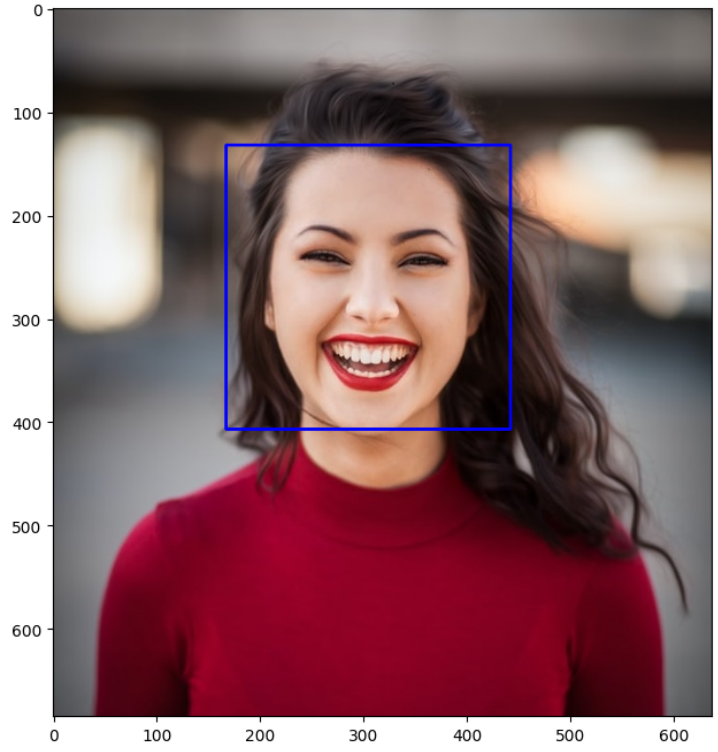
```
#Load the test image and create a greyscale copy of it to be used in the classifiers

base_image = cv2.imread('lady.jpg')
grey = cv2.cvtColor(base_image, cv2.COLOR_BGR2GRAY)
plt.imshow(cv2.cvtColor(base_image, cv2.COLOR_BGR2RGB))
```

```
<matplotlib.image.AxesImage at 0x7ff0f8230e20>
```



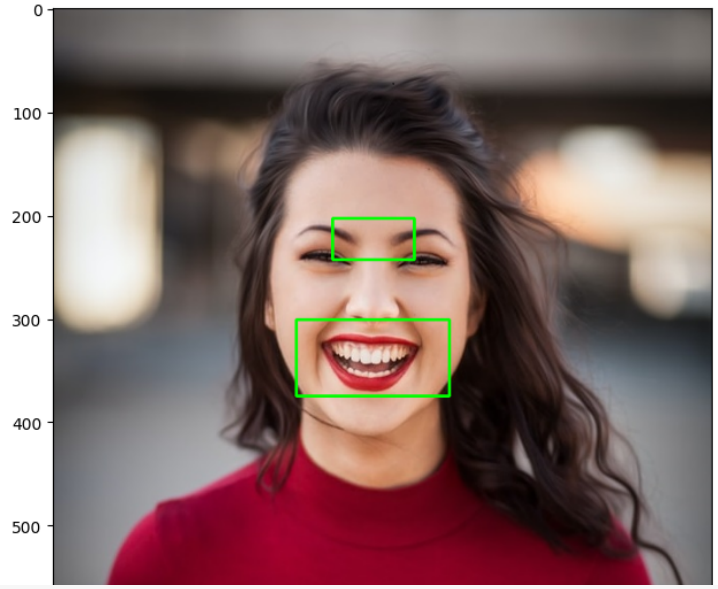

```
#Face detection

#this is a pre-trained face cascade
test_image = cv2.imread('lady.jpg')
face_cascade = cv2.CascadeClassifier('haarcascade_frontalface_default.xml')
faces = face_cascade.detectMultiScale(grey, 1.3, 5)
for (x,y,w,h) in faces:
  cv2.rectangle(test_image,(x,y),(x+w,y+h),(255,0,0),2)
plt.imshow(cv2.cvtColor(test_image, cv2.COLOR_BGR2RGB))
```

```
<matplotlib.image.AxesImage at 0x7ff0f7897fd0>
```




```
#Smile detection

test_image = cv2.imread('lady.jpg')
smile_cascade = cv2.CascadeClassifier('haarcascade_smile.xml')
smiles = smile_cascade.detectMultiScale(grey, 1.3, 20)
for (x,y,w,h) in smiles:
  cv2.rectangle(test_image,(x,y),(x+w,y+h),(0,255,0),2)
plt.imshow(cv2.cvtColor(test_image, cv2.COLOR_BGR2RGB))
```

```
<matplotlib.image.AxesImage at 0x7ff0f77016c0>
```




```
#Eye Detection

test_image = cv2.imread('lady.jpg')
eye_cascade = cv2.CascadeClassifier('haarcascade_eye.xml')
eyes = eye_cascade.detectMultiScale(grey, 1.3, 1)
for (x,y,w,h) in eyes:
  cv2.rectangle(test_image,(x,y),(x+w,y+h),(255,255,255),2)
plt.imshow(cv2.cvtColor(test_image, cv2.COLOR_BGR2RGB))
```

```
<matplotlib.image.AxesImage at 0x7ff0f5b02d10>
```


```
#Putting all together

test_image = cv2.imread('lady.jpg')
for (x,y,w,h) in faces:
  cv2.rectangle(test_image,(x,y),(x+w,y+h),(255,0,0),2)
  for (x_s,y_s,w_s,h_s) in eyes:
    if( (x <= x_s) and (y <= y_s) and ( x+w >= x_s+w_s) and ( y+h >= y_s+h_s)):
      cv2.rectangle(test_image, (x_s,y_s),(x_s+w_s,y_s+h_s),(255,255,255),2)
  for (x_s,y_s,w_s,h_s) in smiles:
    if( (x <= x_s) and (y <= y_s) and ( x+w >= x_s+w_s) and ( y+h >= y_s+h_s)):
      cv2.rectangle(test_image, (x_s,y_s),(x_s+w_s,y_s+h_s),(0,255,0),2)
plt.imshow(cv2.cvtColor(test_image, cv2.COLOR_BGR2RGB))
```

```
<matplotlib.image.AxesImage at 0x7ff0f5001720>
```