**Fire Detection Using OpenCV**

Fire Detection using OpenCV in python Programming. This is a basic program to detect fire using primary/secondary camera of Laptop/pc .

For this mini–project we’ll need three libraries :

* OpenCV
* NumPy
* Matplotlib

We’ll study this project in three steps :

**1. Reading Live video footage :**

Create a variable which will start the video capturing,

live\_Camera = cv2.VideoCapture(0)

^

*The function takes argument 0 or 1 depending upon whether you are using primary camera or external webcam .*

*i.e 0 – primary camera*

*1 – Secondary camera*

Now start reading the footage in infinite while loop ,

while(live\_Camera.isOpened()):

^

*isOpened() checks whether the camera has initialized properly*

ret, frame = live\_Camera.read()

cv2.imshow("Fire Detection",frame)

Now when we’ll have to end or stop the video , Let’s integrate esc button to end the video. But even after pressing the button , we have to release the camera from video capture. So OpenCV provides this functionality.

if cv2.waitKey(10) == 27 :

break

live\_Camera.release()

cv2.destroyAllWindows()

So the overall code looks something like this till now…

import cv2

import numpy as np

import matplotlib.pyplot as plt

live\_Camera = cv2.VideoCapture(0)

while(live\_Camera.isOpened()):

ret, frame = live\_Camera.read()

cv2.imshow("Fire Detection",frame)

if cv2.waitKey(10) == 27 :

break

live\_Camera.release()

cv2.destroyAllWindows()

**2. Using Image processing which will detect Fire**

As we already read the live video. Now let’s just get one single frame from the footage to perform our next operation.

Here we’ll use matplotlib library , and add few line of code below.

img\_RGB = cv2.cvtColor(frame,cv2.COLOR\_BGR2RGB)

^

*Here is one tricky part. OpenCV reads images in BGR format whereas Matplotlib reads image in RGB format. So to get output image in RGB form, we’ll have to convert the frame into RGB from BGR.*

img\_cap = plt.imshow(img\_RGB)

plt.show()

*Code after adding above lines…*

import cv2

import numpy as np

import matplotlib.pyplot as plt

live\_Camera = cv2.VideoCapture(0)

while(live\_Camera.isOpened()):

ret, frame = live\_Camera.read()

cv2.imshow("Fire Detection",frame)

if cv2.waitKey(10) == 27 :

break

img\_RGB = cv2.cvtColor(frame,cv2.COLOR\_BGR2RGB)

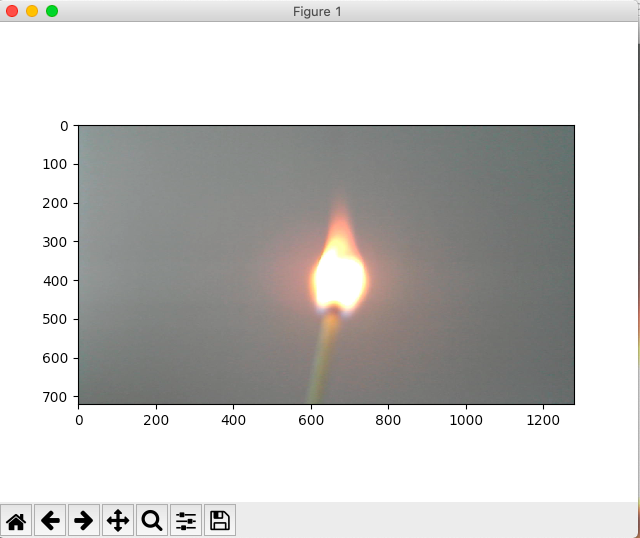
img\_cap = plt.imshow(img\_RGB)

plt.show()

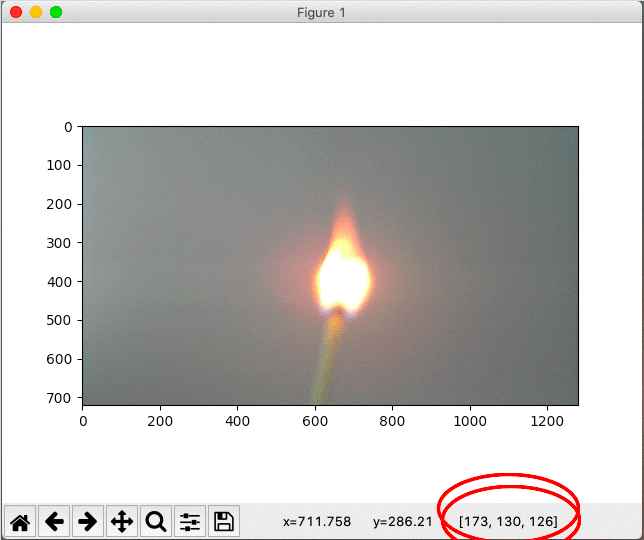
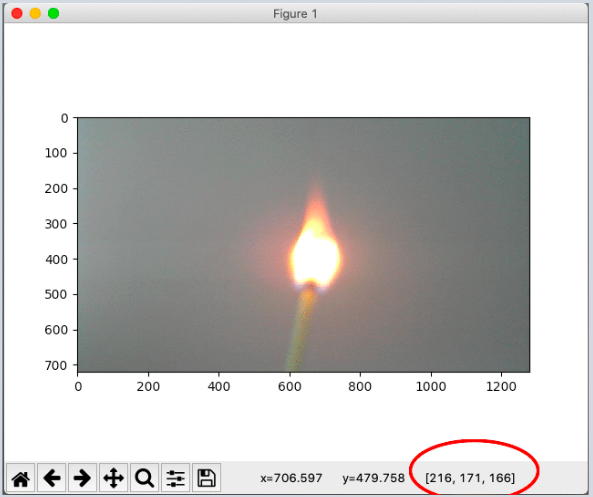
live\_Camera.release()

cv2.destroyAllWindows()

Here I have used Matchsticks as sample to detect fire.



( *This matplotlib windows opens up when you hit the* **esc** *button to end the footage*)

You’ll notice one thing that when hovering your cursor over the image, RBG values ( i.e. shown in the red circles on image ) changes.

Here, we’ll choose two such values where our region (i.e. flames of the matchstick) lies between those two values. ( i.e. lower\_region & upper\_region)

After selecting those two values, convert them into a NumPy array and store it in variable respectively.

lower\_bound = np.array([11,33,111])

upper\_bound = np.array([90,255,255])

Note : The colour scheme that we got through matplotlib is in RGB format. So we’ll have to convert it into BGR format so that OpenCV can read it.

Example : Assume we got [ 111, 33, 11 ] through the image.

^ ^ ^

R. G. B.

So we’ll write as [ 11, 33, 111 ]

^ ^ ^

B G R

Now the main part comes, i.e. Image Processing

Step 1 : Appling smoothing technique ( Gaussian Blur )

frame\_smooth = cv2.GaussianBlur(frame,(7,7),0)

*GaussianBlur(src, ksize, sigmaX)*

This method accepts the following parameters −

* **src** − object representing the source (input image) for this operation.
* **ksize** − A **Size** object representing the size of the kernel.
* **sigmaX** − A variable of the type double representing the Gaussian kernel standard deviation in X direction.

Step 2 : Masking Technique ( NumPy )

*Create an array of ‘zeros’ with default size same as our size of the frame from live footage .*

mask = np.zeros\_like(frame)

*Now assign the mask with white colour in BGR format.*

mask[0:720, 0:1280] = [255,255,255]

^

Define the size of *Region of interest* according to your preference.

But I’ll prefer you to keep the ROI same as the size of your window.

**Till now we have defined the colour of our flame that we want to detect , and created mask image of white colour (i.e. all 1’s)**

Step 3 : ROI operation

img\_roi = cv2.bitwise\_and(frame\_smooth, mask)

^

We want to only detect the flames of the fire. So img1 = frame\_smooth gives image of the current frame, and img2 = mask is array of value of colour black ,it's value is 0 in OpenCV.

But we have changed the colour of our mask to white (i.e. 1).

According to the logic table of bitwise\_and, if both the inputs are high, the output is high. Therefore only the pixels that lie in the region of mask are shown as the output.

Step 4 : Define a Threshold

Now that we have region of interest to track the colour of flame that we defined earlier,

*We first convert the ROI to HSV format.*

frame\_hsv = cv2.cvtColor(img\_roi,cv2.COLOR\_BGR2HSV)

Note :

*Why do we convert from RBG/BGR to HSV ?*

*Because the R, G, and B components of an object’s colour in a digital image are all correlated with the amount of light hitting the object, and therefore with each other, image descriptions in terms of those components make object discrimination difficult. Descriptions in terms of hue/lightness/chroma or hue/lightness/saturation are often more relevant.*

Now the final step is to detect the flames. Keeping the source image as Live Video and limits as lower\_bound & upper\_bound defined earlier.

image\_binary = cv2.inRange(frame\_hsv, lower\_bound, upper\_bound)

*The code till now looks like this…*

import cv2

import numpy as np

import matplotlib.pyplot as plt

live\_Camera = cv2.VideoCapture(0)

lower\_bound = np.array([11,33,111])

upper\_bound = np.array([90,255,255])

while(live\_Camera.isOpened()):

ret, frame = live\_Camera.read()

frame = cv2.resize(frame,(1280,720))

frame = cv2.flip(frame,1)

frame\_smooth = cv2.GaussianBlur(frame,(7,7),0)

mask = np.zeros\_like(frame)

mask[0:720, 0:1280] = [255,255,255]

img\_roi = cv2.bitwise\_and(frame\_smooth, mask)

frame\_hsv = cv2.cvtColor(img\_roi,cv2.COLOR\_BGR2HSV)

image\_binary = cv2.inRange(frame\_hsv, lower\_bound, upper\_bound)

cv2.imshow("Fire Detection",image\_binary)

if cv2.waitKey(10) == 27 :

break

live\_Camera.release()

cv2.destroyAllWindows()

*So the output will show only white scale image of the flames of fire. Here is an example below…*



3. Final Modification to the code

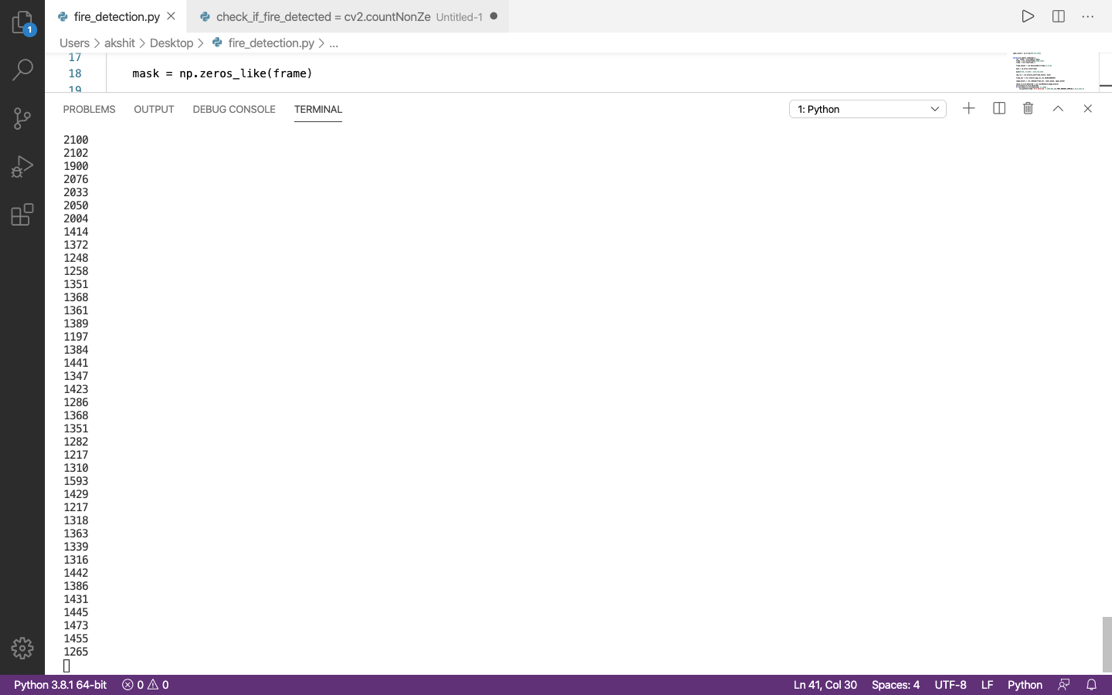
check\_if\_fire\_detected = cv2.countNonZero(image\_binary)

^

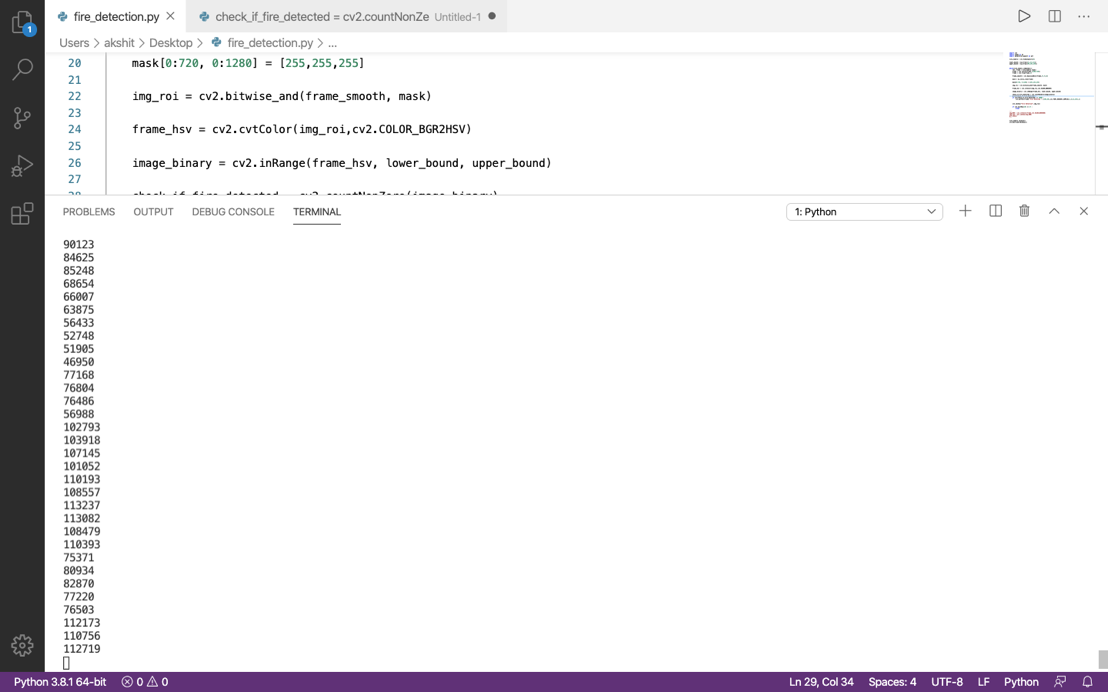
Returns number of non-zero pixels

Now print this ‘ check\_if\_fire\_detected ‘,

print(check\_if\_fire\_detected)



This is returning some integer values. And when flames are detected the values increased rapidly.



So, lets write a condition here such that :

if int(check\_if\_fire\_detected) >= 20000 :

cv2.putText(frame,"FireDetected!",(300,60),cv2.FONT\_HERSHEY\_COMPLEX,3,(0,0,255),2)

Finally the code is done !

import cv2

import numpy as np

import matplotlib.pyplot as plt

live\_Camera = cv2.VideoCapture(0)

lower\_bound = np.array([11,33,111])

upper\_bound = np.array([90,255,255])

while(live\_Camera.isOpened()):

ret, frame = live\_Camera.read()

frame = cv2.resize(frame,(1280,720))

frame = cv2.flip(frame,1)

frame\_smooth = cv2.GaussianBlur(frame,(7,7),0)

mask = np.zeros\_like(frame)

mask[0:720, 0:1280] = [255,255,255]

img\_roi = cv2.bitwise\_and(frame\_smooth, mask)

frame\_hsv = cv2.cvtColor(img\_roi,cv2.COLOR\_BGR2HSV)

image\_binary = cv2.inRange(frame\_hsv, lower\_bound, upper\_bound)

check\_if\_fire\_detected = cv2.countNonZero(image\_binary)

if int(check\_if\_fire\_detected) >= 20000 :

cv2.putText(frame,"Fire Detected !",(300,60),cv2.FONT\_HERSHEY\_COMPLEX,3,(0,0,255),2)

cv2.imshow("Fire Detection",frame)

if cv2.waitKey(10) == 27 :

break

live\_Camera.release()

cv2.destroyAllWindows()