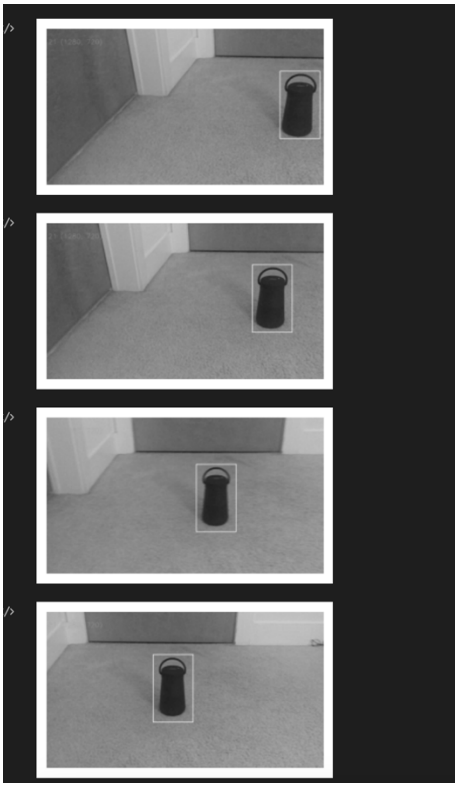
Computer Vision

3rd Assignment

Name: Srinivas Narne ***Panther Id: 2705961***

1. Capture 10-second video footage using the camera. The footage should be taken with the camera in hand, and you need to pan the camera slightly from left-right or right-left during the 10-sec duration. Pick any image frame from the 10-sec video footage. Pick a region of interest corresponding to an object in the image. Crop this region from the image. Then use this cropped region to compare with randomly picked 10 images in the dataset of 10-sec video frames to see if there is a match for the object in the scenes from the 10 images. For comparison, use the sum of squared differences (SSD) or normalized correlation.

Cropped image in the video

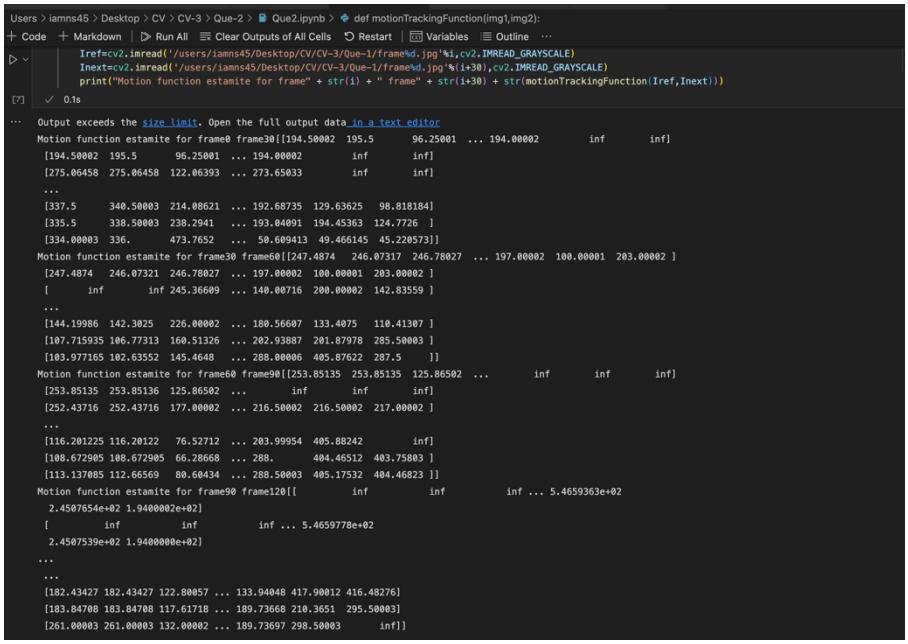


Output

The shape of the Speaker is perfectly identified in the rest of the frames in the video. I have shown the output with a rectangle identifying the box in other frames.

---

2. Implement the motion tracking equation from fundamental principles. Select any 2 consecutive frames from the set from problem 1 and compute the motion function estimates. Conduct image registration to realign the frames. Repeat the test for all consecutive pairs of frames in the video.

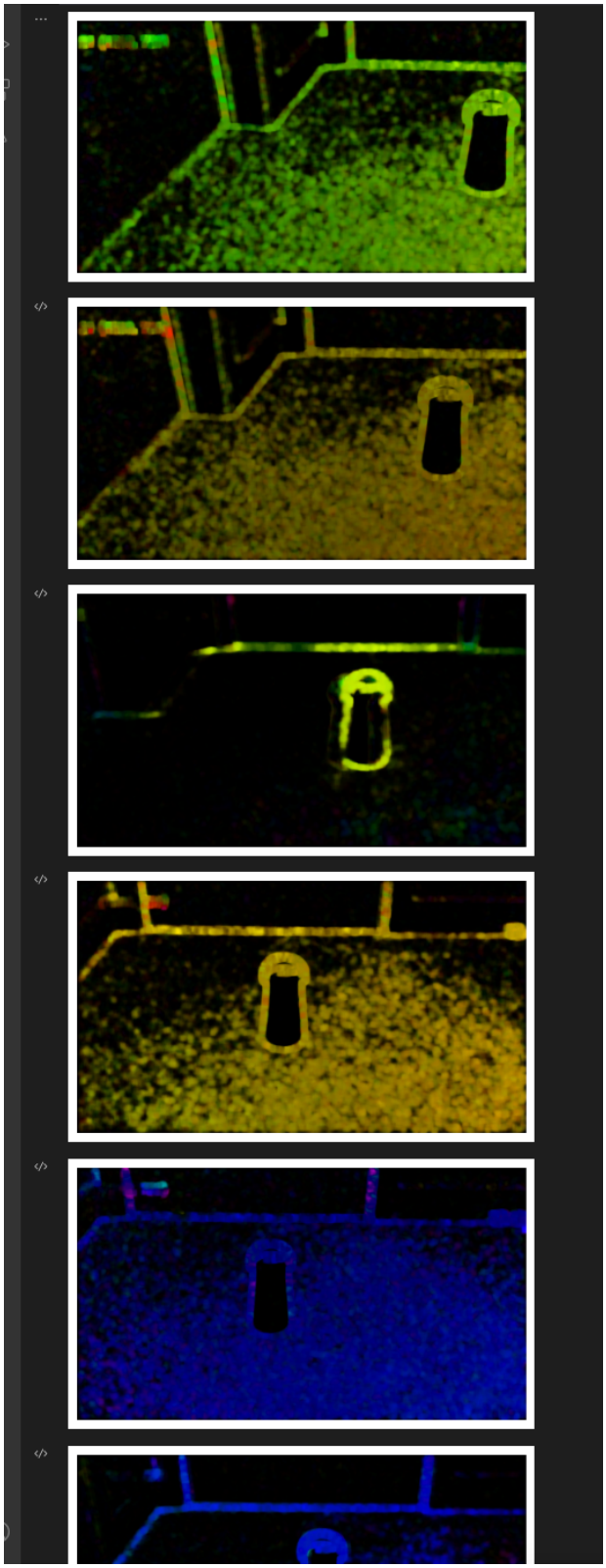


```
Users > iamns45 > Desktop > CV > CV-3 > Que-2 > Que2.ipynb > def motionTrackingFunction(img1,img2):
+ Code + Markdown | Run All Clear Outputs of All Cells Restart | Variables Outline ···
    Iref=cv2.imread('/users/iamns45/Desktop/CV/CV-3/Que-1/frame%d.jpg'%i,cv2.IMREAD_GRAYSCALE)
    Inext=cv2.imread('/users/iamns45/Desktop/CV/CV-3/Que-1/frame%d.jpg'%(i+30),cv2.IMREAD_GRAYSCALE)
    print("Motion function estamite for frame" + str(i) + " frame" + str(i+30) + str(motionTrackingFunction(Iref,Inext)))
[7]  ✓ 0.1s
···  Output exceeds the size limit. Open the full output data in a text editor
Motion function estamite for frame0 frame30[[194.50002  195.5       96.25001  ... 194.00002        inf       inf]
 [194.50002  195.5       96.25001  ... 194.00002        inf       inf]
 [275.06458  275.06458  122.06393  ... 273.65033        inf       inf]
 ...
 [337.5      340.50003  214.08621  ... 192.68735  129.63625   98.818184]
 [335.5      338.50003  238.2941   ... 193.04091  194.45363  124.7726  ]
 [334.00003  336.       473.7652   ...  50.609413  49.466145  45.220573]]
Motion function estamite for frame30 frame60[[247.4874   246.07317  246.78027  ... 197.00002  100.00001  203.00002 ]
 [247.4874   246.07321  246.78027  ... 197.00002  100.00001  203.00002 ]
 [       inf        inf 245.36609  ... 140.00716  200.00002  142.83559 ]
 ...
 [144.19986  142.3025   226.00002  ... 180.56607  133.4075   110.41307 ]
 [107.715935 106.77313  160.51326  ... 202.93887  201.87978  285.50003 ]
 [103.977165 102.63552  145.4648   ... 288.00006  405.87622  287.5     ]]
Motion function estamite for frame60 frame90[[253.85135  253.85135  125.86502  ...        inf       inf       inf]
 [253.85135  253.85136  125.86502  ...        inf       inf       inf]
 [252.43716  252.43716  177.00002  ... 216.50002  216.50002  217.00002 ]
 ...
 [116.201225 116.20122   76.52712  ... 203.99954  405.88242        inf]
 [108.672905 108.672905  66.28668  ... 288.       404.46512  403.75803 ]
 [113.137085 112.66569   80.60434  ... 288.50003  405.17532  404.46823 ]]
Motion function estamite for frame90 frame120[[          inf          inf          inf ... 5.4659363e+02
  2.4507654e+02 1.9400002e+02]
 [          inf          inf          inf ... 5.4659778e+02
  2.4507539e+02 1.9400000e+02]
 ...
 ...
 [182.43427 182.43427 122.80057 ... 133.94048 417.90012 416.48276]
 [183.84708 183.84708 117.61718 ... 189.73668 210.3651  295.50003]
 [261.00003 261.00003 132.00002 ... 189.73697 298.50003       inf]]
```

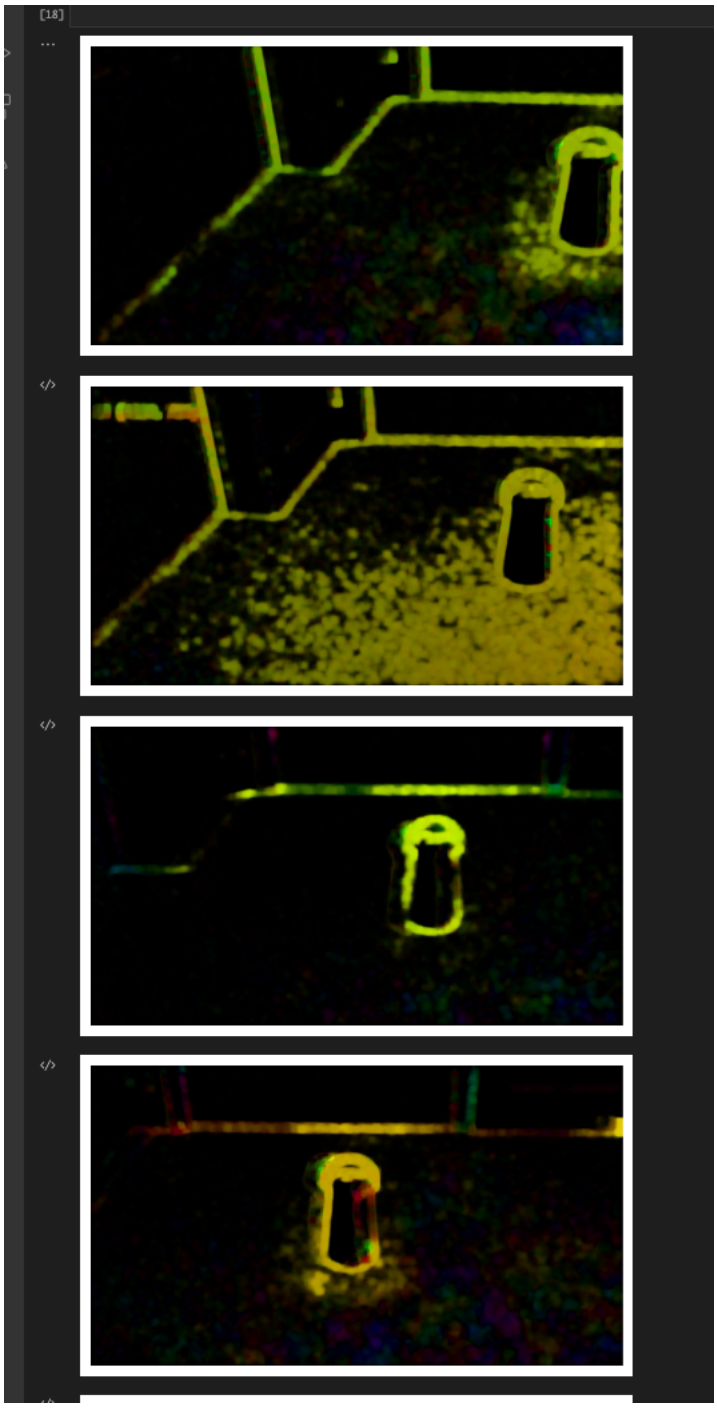
The motion function estimate for frames I , i+30

3. For the video (problem 1) you have taken, plot the optical flow vectors on each frame.

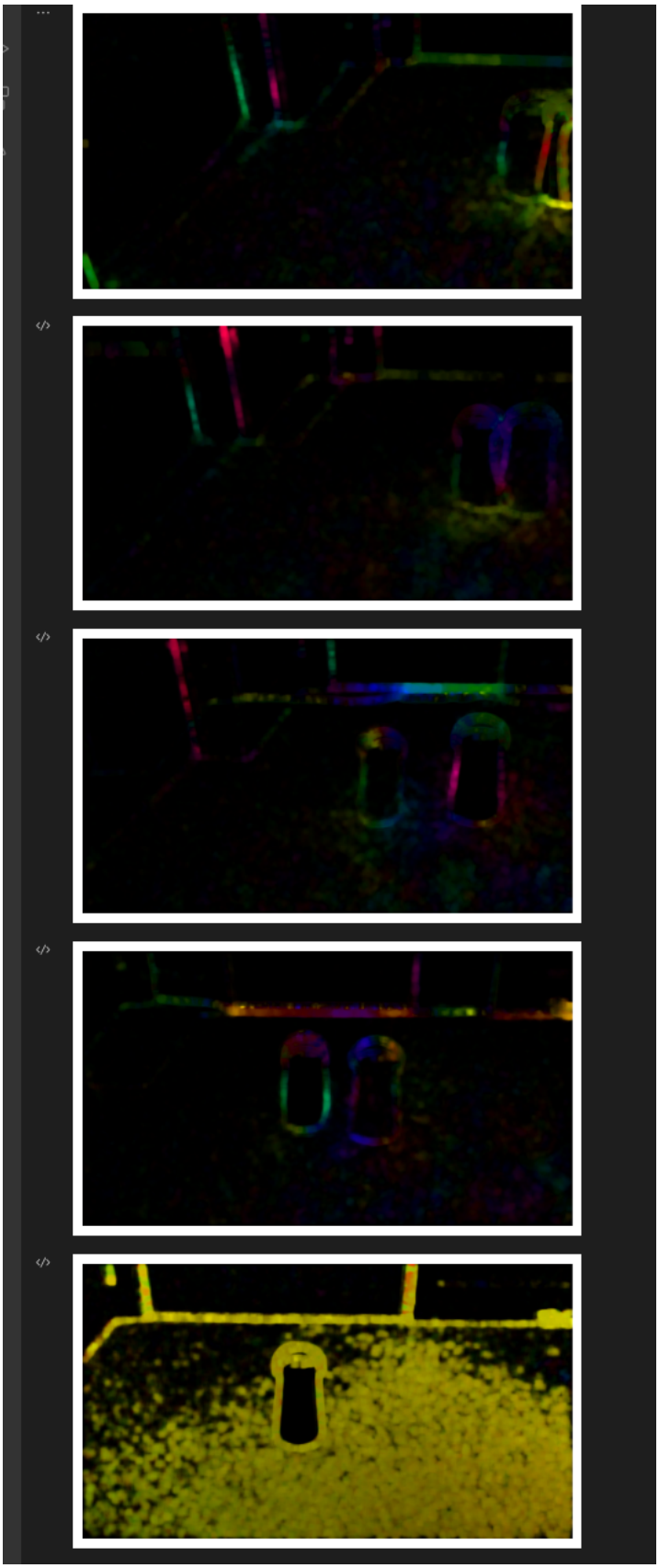
(i) treating every previous frame as a reference frame



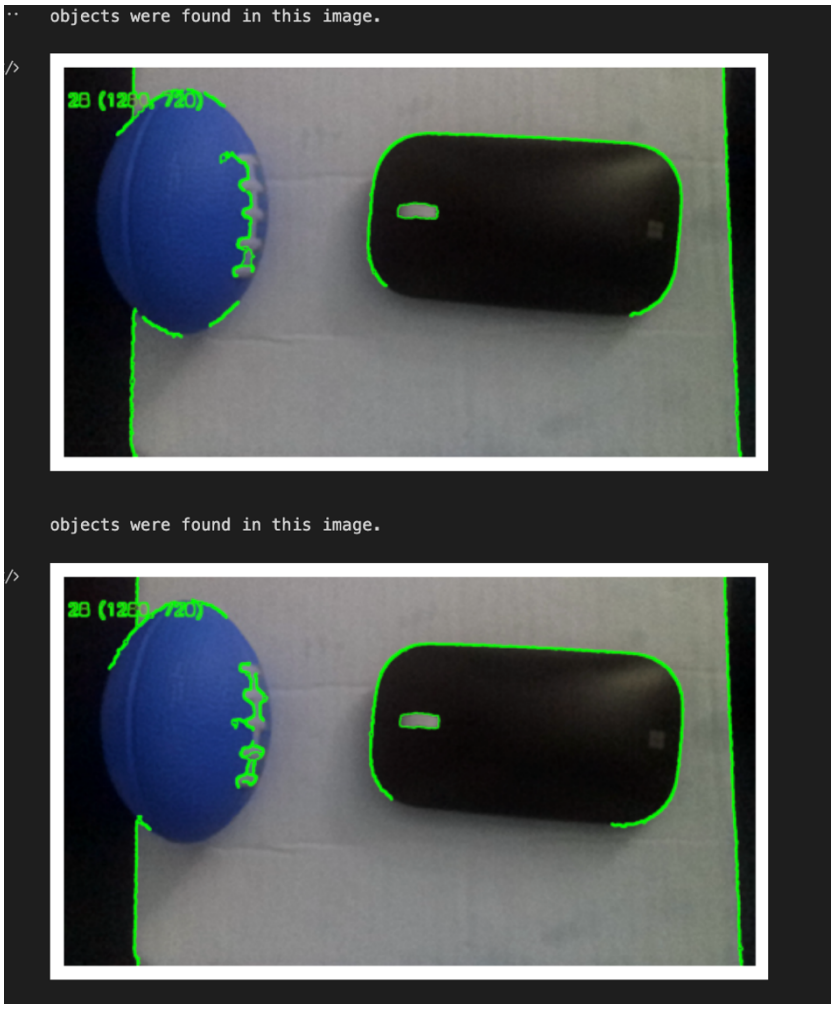
(ii) treating every 11th frame as a reference frame



(iii) treating every 31st frame as a reference frame

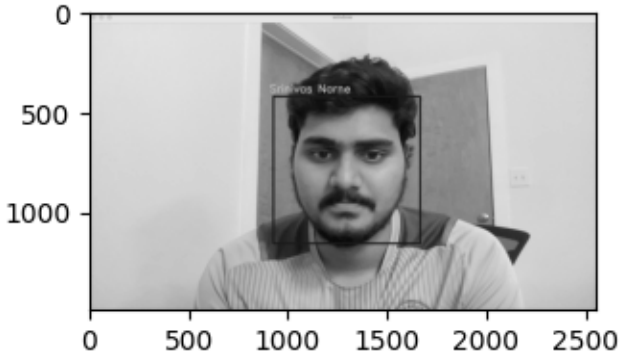


4. Implement a feature-based object detection application (from scratch) for detecting an object of your choice. Test it for at least 2 differently-looking objects. Validate your results by testing against built-in object detection functions/code in MATLAB/OpenCV



Identified Objects

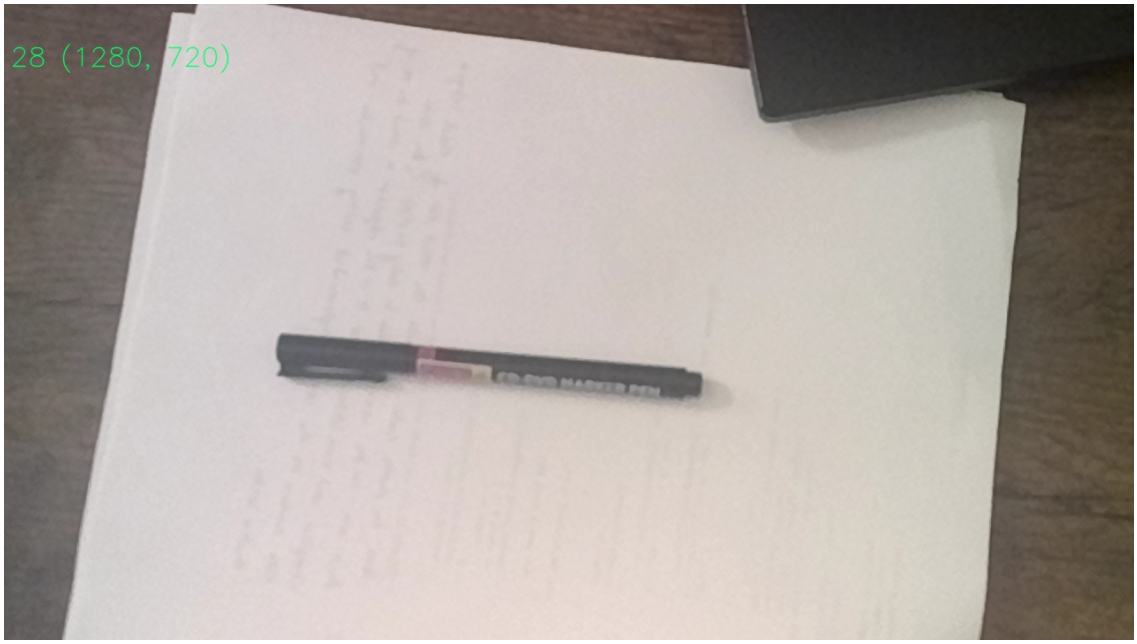
5. Implement a real-time face tracking application that will detect as many faces there are with a scene, and identify the person's facial region (draw a bounding box) whose is sought for by the user (you must ask for a person in your application and it should show a bounding box over the person of interest). Validate at least 20 times and present the recognition performance metrics (accuracy, precision, recall and Intersection over Union (IoU)).

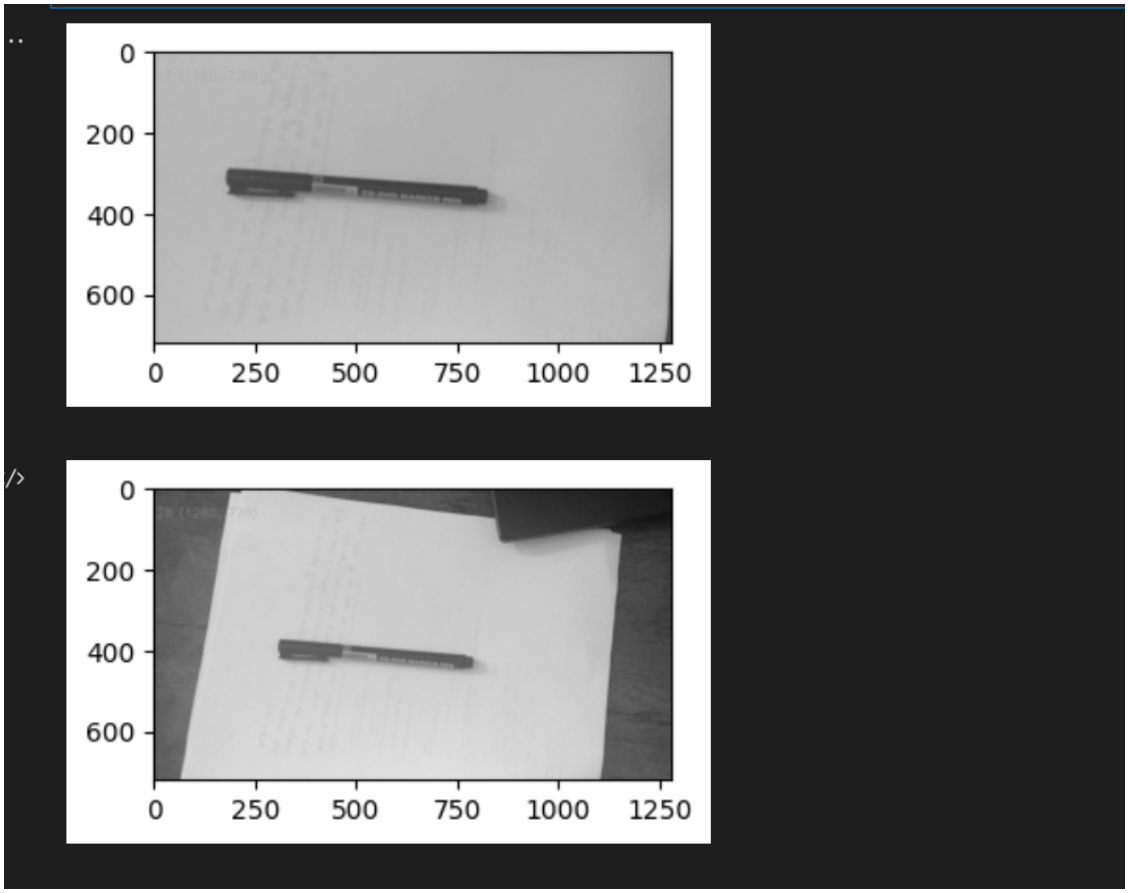


Multiple face detection sample image 1

---

6. Fix a marker on a wall or a flat vertical surface. From a distance D, keeping the camera stationed static (not handheld and mounted on a tripod or placed on a flat surface), capture an image such that the marker is registered. Then translate the camera by T units along the axis parallel to the ground (horizontal) and then capture another image, with the marker being registered. Compute D using disparity-based depth estimation in stereo-vision theory. (Note: you can pick any value for D, and T.





Output

The Disparity based depth estimation is 938.9388810056869mm.

GitHub Link-

https://github.com/Iamns45/Computer-Vision/tree/main/CV-3