COMPUTER VISION
Assignment-2

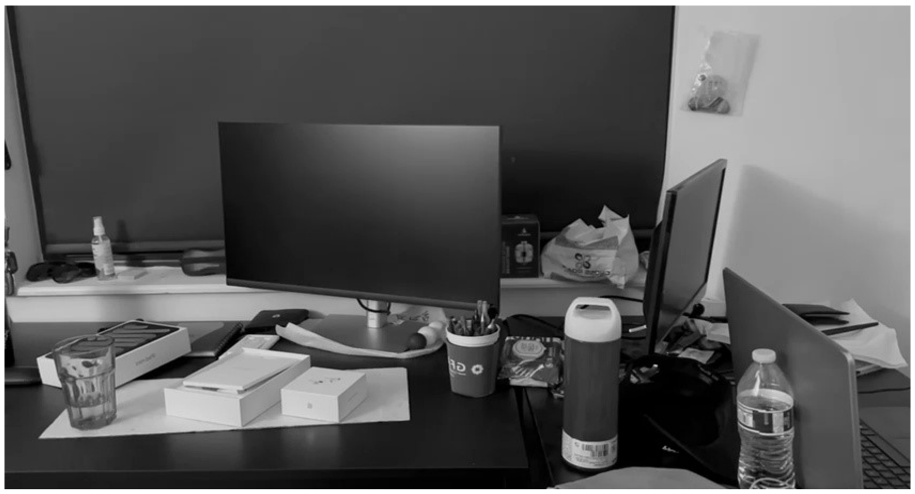
Name: Srinivas Narne **Panther ID:2705961**

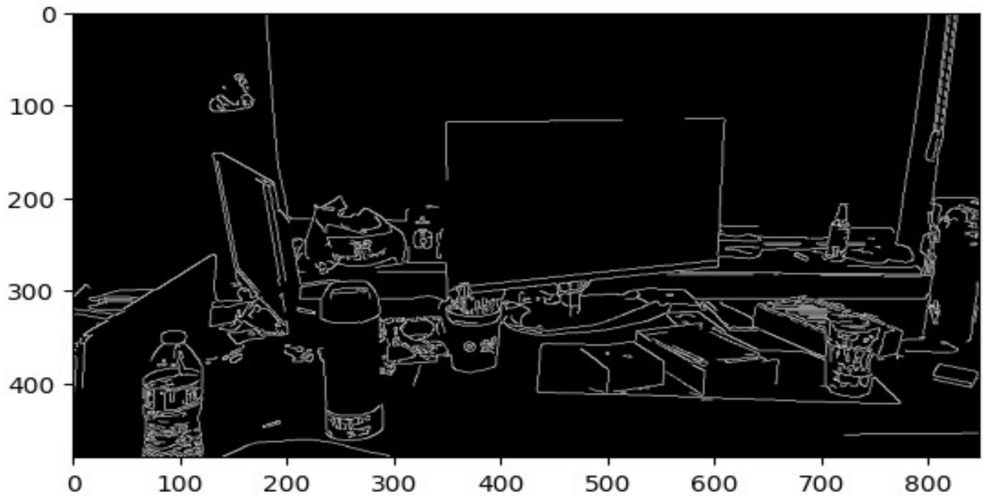
Question1:

Canny Detection:

The amount of data that needs to be processed can be significantly decreased while still obtaining valuable structural information from various visual objects with the help of the Canny edge detection technology. In various computer vision systems, it is commonly employed.

Consider the image given for canny edge detection.

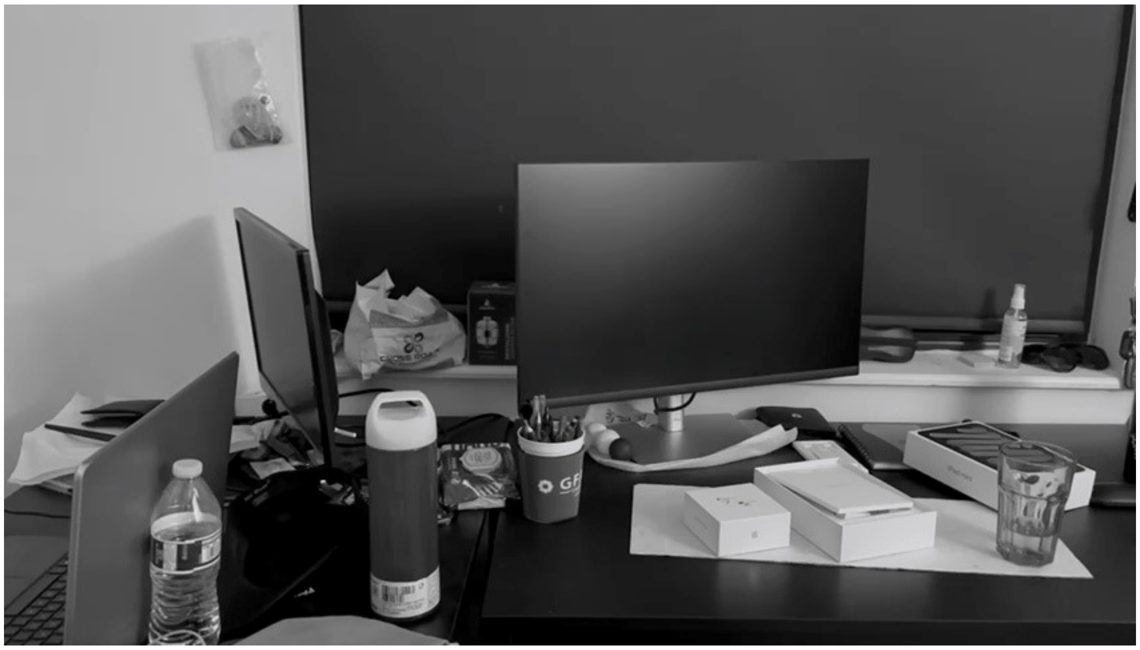


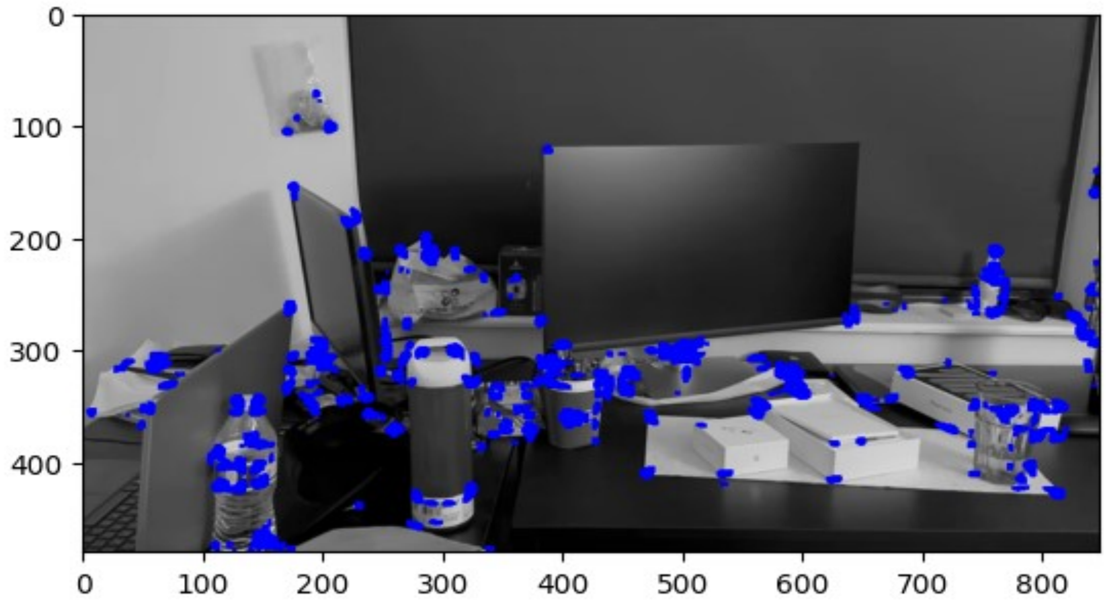


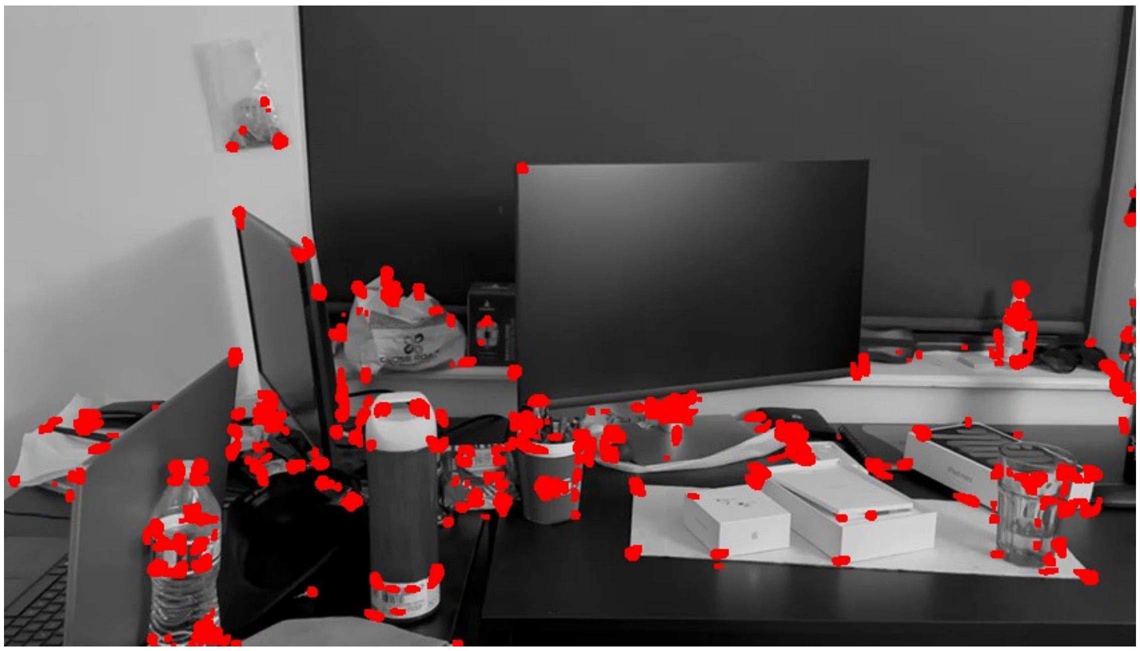
Harris Corner Detection:

The Harris corner detector is a corner detection operator that is extensively used in computer vision algorithms to extract corners and infer attributes of an image.

Taken a video and picked two images from it and placed them in the folder. Harris corner detection. Below is the output on execution.







b. Pick another image frame from the set with the same object in view. Find all corresponding points of the object under consideration between these two images. Find the Homography matrix between the images

The Canny edge detection algorithm is composed of 5 steps:

- Noise reduction
- Gradient calculation
- Non-maximum suppression
- Double threshold
- Edge Tracking by Hysteresis.

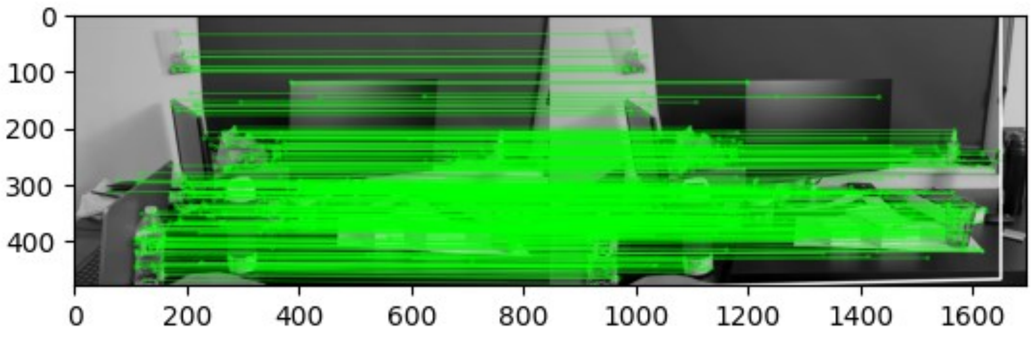


Image Coordinates Detection:

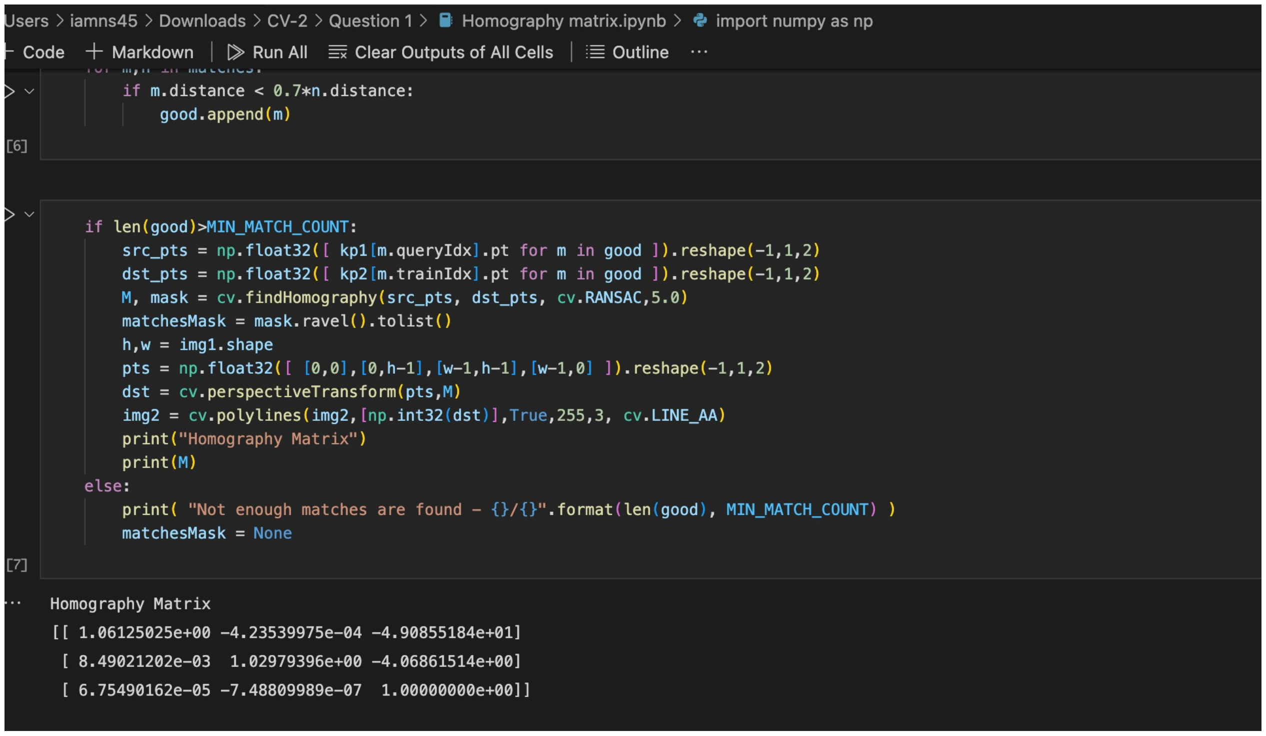
Another image, img2C.png, is taken, and the Image_coordinates Detection. ipynb script is used to get 5 corners of the monitor object used in the image. The following is the output after execution.

[275,319],[796,313],[769,575],[251,535]
[651,446],[968,442],[996,591],[657,598]

I used these coordinates and run the script Homography matrix. ipynb to get output. One execution of the below is the output.

Homography Matrix:

[[1.06125025e+00 -4.23539975e-04 -4.90855184e+01]
 [8.49021202e-03 1.02979396e+00 -4.06861514e+00]
 [6.75490162e-05 -7.48809989e-07 1.00000000e+00]]

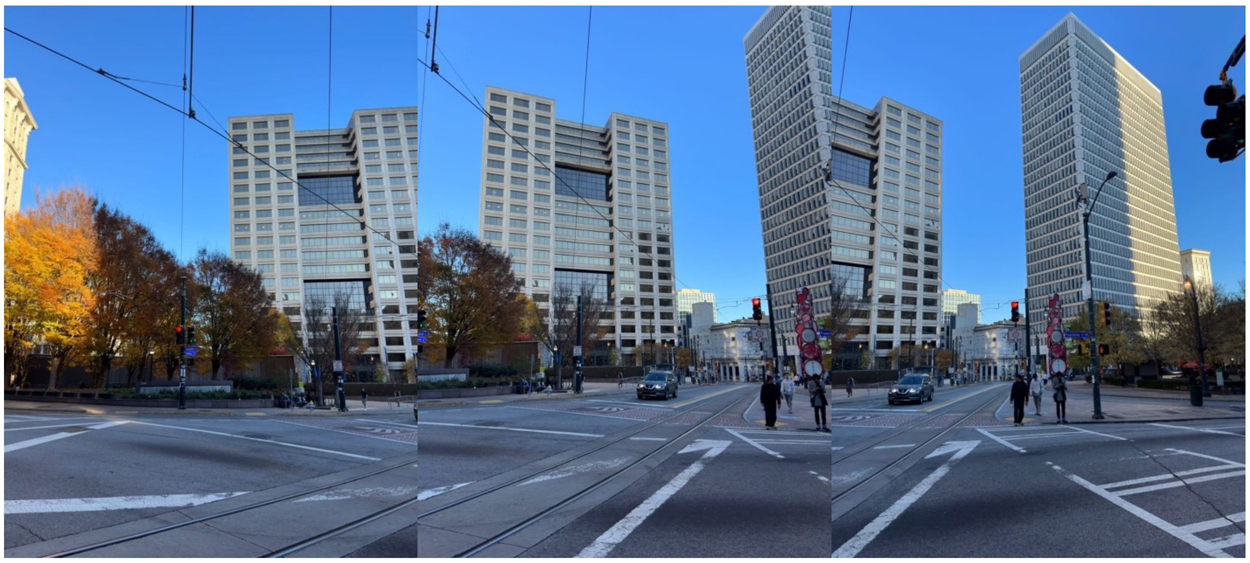


```
Users > iamns45 > Downloads > CV-2 > Question 1 > Homography matrix.ipynb > import numpy as np
Code  Markdown  Run All  Clear Outputs of All Cells  Outline  ...

        if m.distance < 0.7*n.distance:
            good.append(m)
[6]

    if len(good)>MIN_MATCH_COUNT:
        src_pts = np.float32([ kp1[m.queryIdx].pt for m in good ]).reshape(-1,1,2)
        dst_pts = np.float32([ kp2[m.trainIdx].pt for m in good ]).reshape(-1,1,2)
        M, mask = cv.findHomography(src_pts, dst_pts, cv.RANSAC,5.0)
        matchesMask = mask.ravel().tolist()
        h,w = img1.shape
        pts = np.float32([ [0,0],[0,h-1],[w-1,h-1],[w-1,0] ]).reshape(-1,1,2)
        dst = cv.perspectiveTransform(pts,M)
        img2 = cv.polylines(img2,[np.int32(dst)],True,255,3, cv.LINE_AA)
        print("Homography Matrix")
        print(M)
    else:
        print( "Not enough matches are found - {}/{}".format(len(good), MIN_MATCH_COUNT) )
        matchesMask = None
[7]

... Homography Matrix
    [[ 1.06125025e+00 -4.23539975e-04 -4.90855184e+01]
     [ 8.49021202e-03  1.02979396e+00 -4.06861514e+00]
     [ 6.75490162e-05 -7.48809989e-07  1.00000000e+00]]
```

Question2:

Performed the image stitching operation on 5 sets of buildings/images with 3 images per set to form the stitching.

One of the datasets is as below.

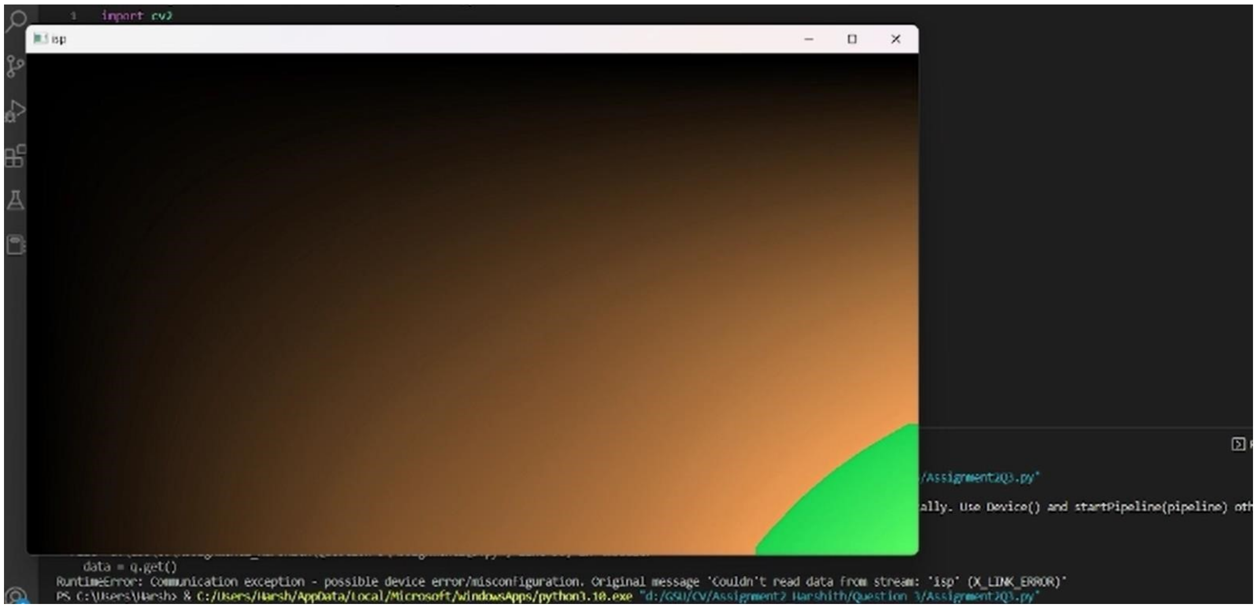


The result of image stitching for the above is

Question3:

In question 3, the script **Que3.ipynb** file **integral image function** calculates the integral values of the image, which is custom written. The output has been saved.

Sample output:



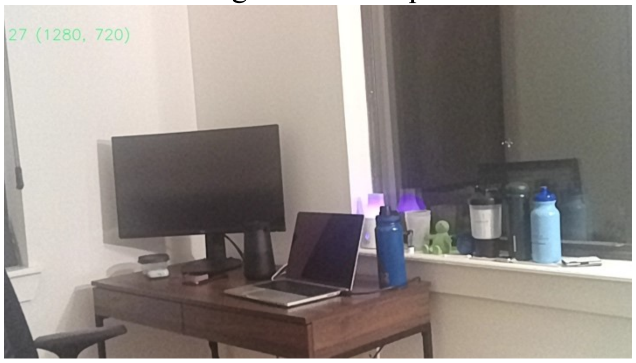
```
import cv2
```

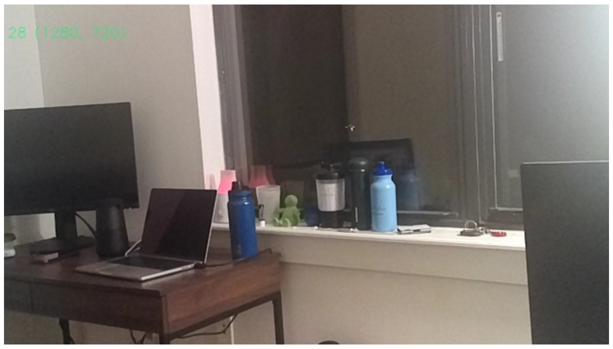
```
/Assignment2Q3.py"
ally. Use Device() and startPipeline(pipeline) oth
    data = q.get()
RuntimeError: Communication exception - possible device error/misconfiguration. Original message 'Couldn't read data from stream: 'isp' (X_LINK_ERROR)'
PS C:\Users\Harsh> & C:/Users/Harsh/AppData/Local/Microsoft/WindowsApps/python3.10.exe "d:/GSU/CV/Assignment2 Harshith/Question 3/Assignment2Q3.py"
```

Question4:

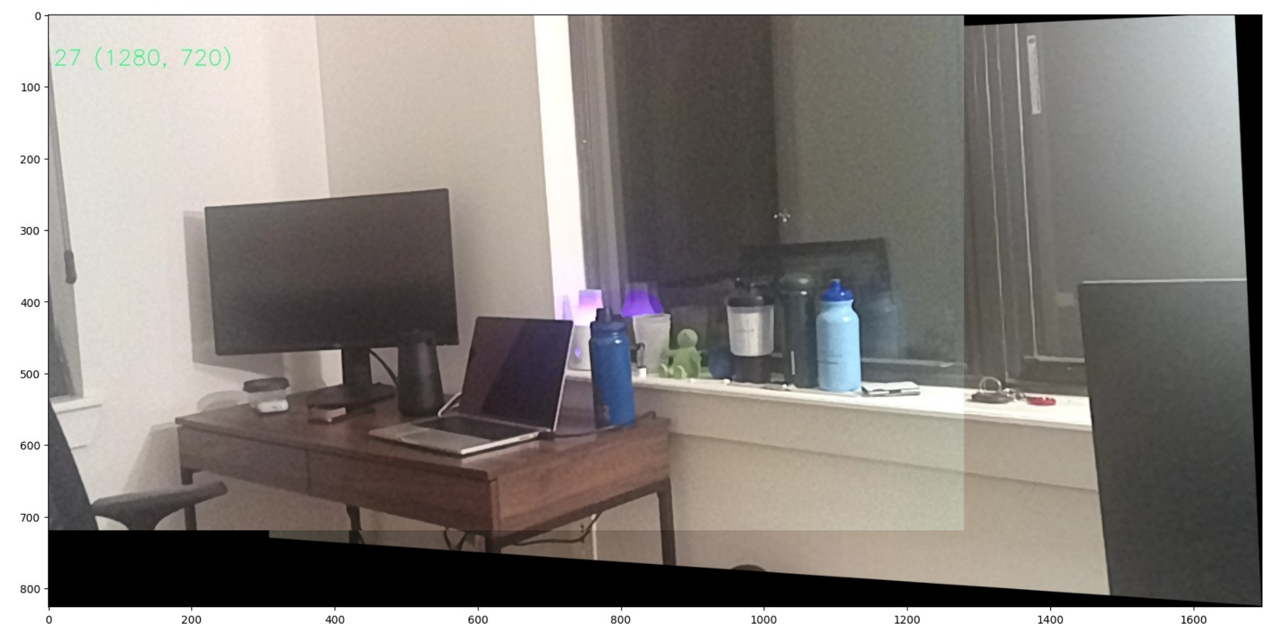
Img_sift.ipynb is the source code file used to execute task 4. It gives the stitched image sift_Result.png. The main logic is to find the main feature points using the cv2 sift algorithm in Main_Ponts_Func(). Then, Match_The_Key_Points_Func() uses the KNN algorithm to find a nearest possible match between images. Then, use Find_Homography() method to find homography and use it to stitch the image using cv2.WarpPresepective() function to get the result image.

Below are the images used as input to the file.



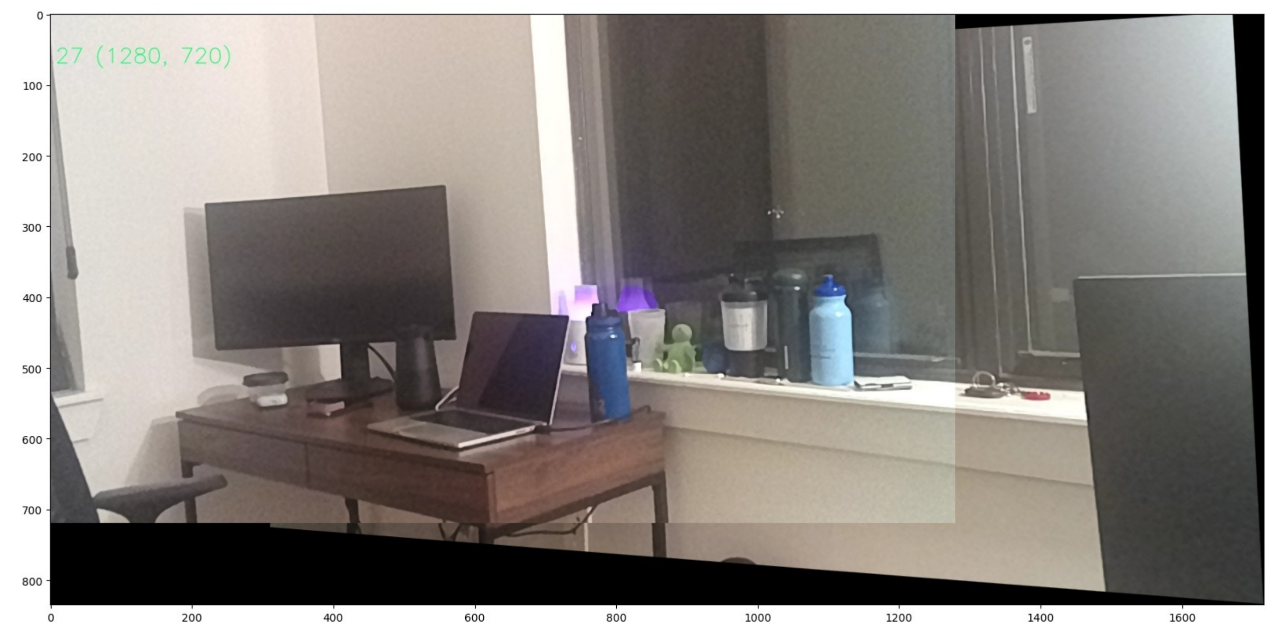


The output file is as below:



Question5:

In question5 folder Que5.ipynb, the script is the same as to sift script, but the Main_Ponts_Func() uses cv2 orb feature detection.



Github Link:

https://github.com/Iamns45/Computer-Vision/tree/main/CV-2