**AGV Task 6**

**By Uddipto Mandal (24IM10065)**

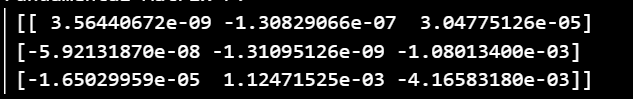
**Sources:**

* <https://www.youtube.com/watch?v=4Fqt2Nk2lhY&list=PLgnQpQtFTOGSO8HC48K9sPuNliY1qxzV9>
* <https://courses.cs.duke.edu/fall13/compsci527/notes/reconstruction.pdf>
* <https://www.youtube.com/watch?v=dUDMQ6dwWDA> (and some other videos in his series)
* Chat gpt
* <https://www.youtube.com/watch?v=5LrAhSHNIJU>
* <https://www.youtube.com/watch?v=6hr6xpOU-uw>

**2.1**

**Implementation of the 8-point algorithm:**

Given 2 images, we can determine relative orientation of image 1 wrt image 2 using this algorithm.

 :This is the fundamental matrix.



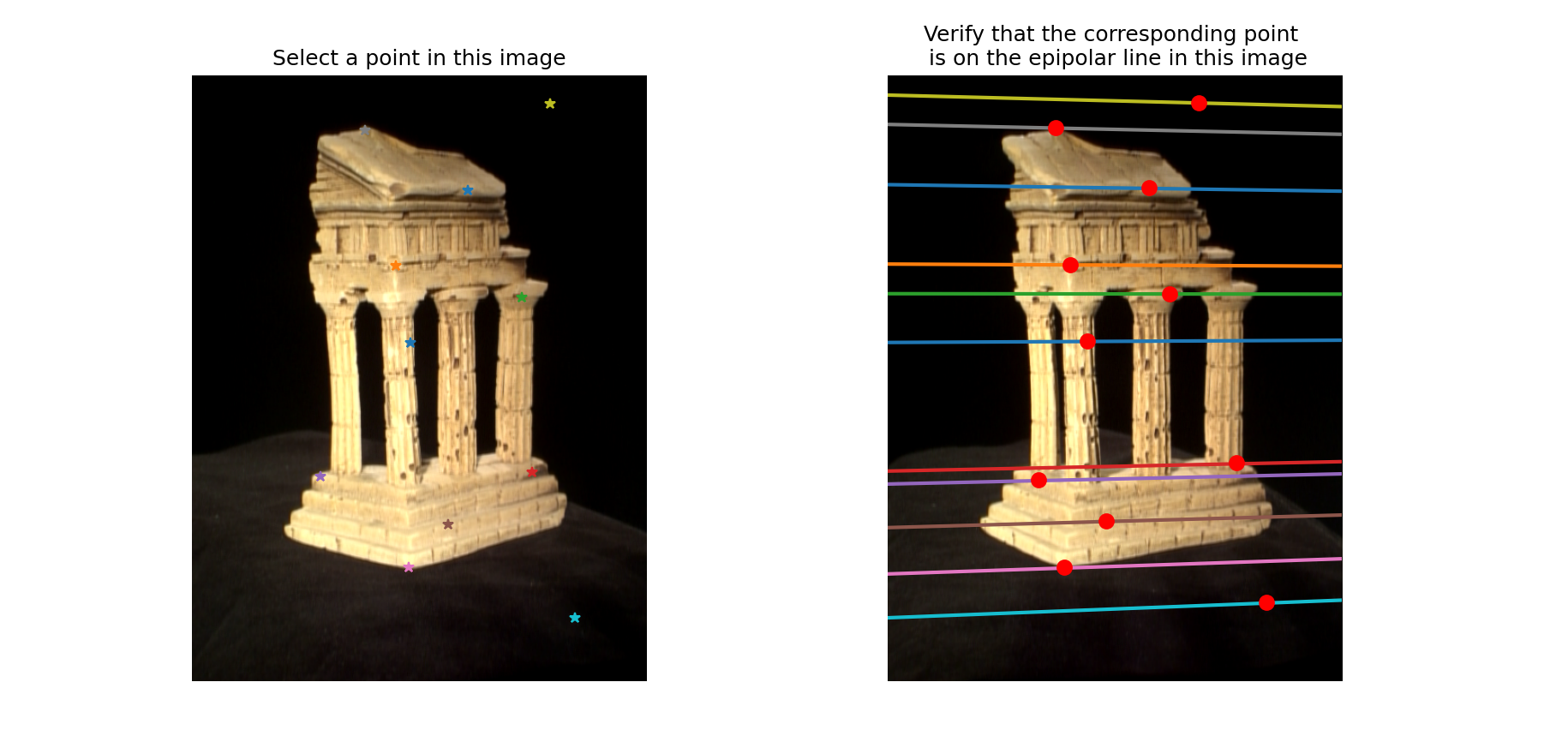
Here we have drawn the epipolar lines for each of the points. This means that knowing a point in image 1, the corresponding point in image 2 lies along the line drawn called the epipolar line of that point. We see that not all the epipolar lines are parallel, but they all seem to converge at a point. This point is called the epipole, which is the projection of the camera center of one camera onto the image plane of the other camera.

Remark: I was getting an error when using refineF helper function, for which I have not used it, but commented out that line.

**2.2**

**Finding epipolar correspondences**

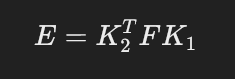
Once we generated the epipolar lines, we know that the corresponding point in image 2 must lie on that line. For 1 point in image 1, we go through all the points in its epipolar line in image 2 and compare similarity between the points. If the similarity is the maximum at a certain point wrt all the other points on the epipolar line, we consider that to be the corresponding line. In my code, I have used the one mentioned in the problem statement itself, which is to take a small window, of size 5x5 around the point in img 1, and compare this window to the corresponding window of the points in epipolar line. I had found the ssd(sum of squared difference, or Euclidean distance) between the 2 windows, and the point in the epipolar line with least ssd has been considered the corresponding epipolar point.



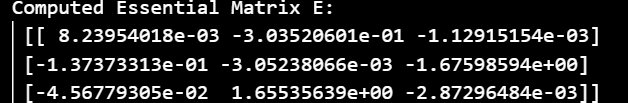
This is the output I got using epipolarMatchGUI from the helper functions. It is coming pretty accurate except for the green point in image 1. I think it is inaccurate at that point since on its epipolar line, there is another point which is very similar to that point, and the SSD was not able to differentiate between the 2 contenders, and it gave the wrong corresponding point. This problem is not coming every time I run the program, only certain times. Apart from that point, the other corresponding points came quite accurately.

**2.3**

**Computing essential matrix**



Where E is essential matrix, F is fundamental matrix, K1 is intrinsic matrix for 1st camera, K2 is intrinsic matrix for 2nd camera. Intrinsic matrices give information on how a camera maps 3d image onto 2d plane assuming the camera sits at (0,0,0) and has orientation (0,0,0). It has some parameters like: camera constant, scale diff in x, y, focal lengths, principal point, etc. The essential matrix relates the normalized image coordinates for corresponding points in 2 calibrated cameras, whereas Fundamental matrix relates coordinates for uncalibrated cameras.

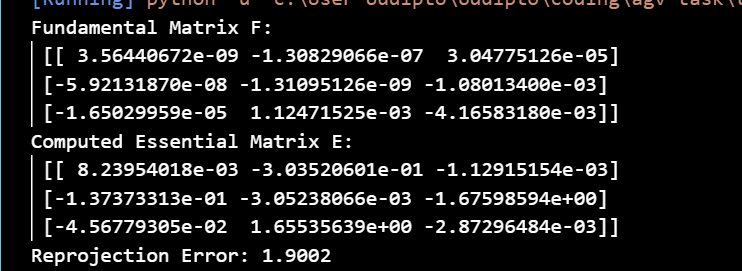


Here is the output for the temple images.

**2.4**

**Implementing triangulation**

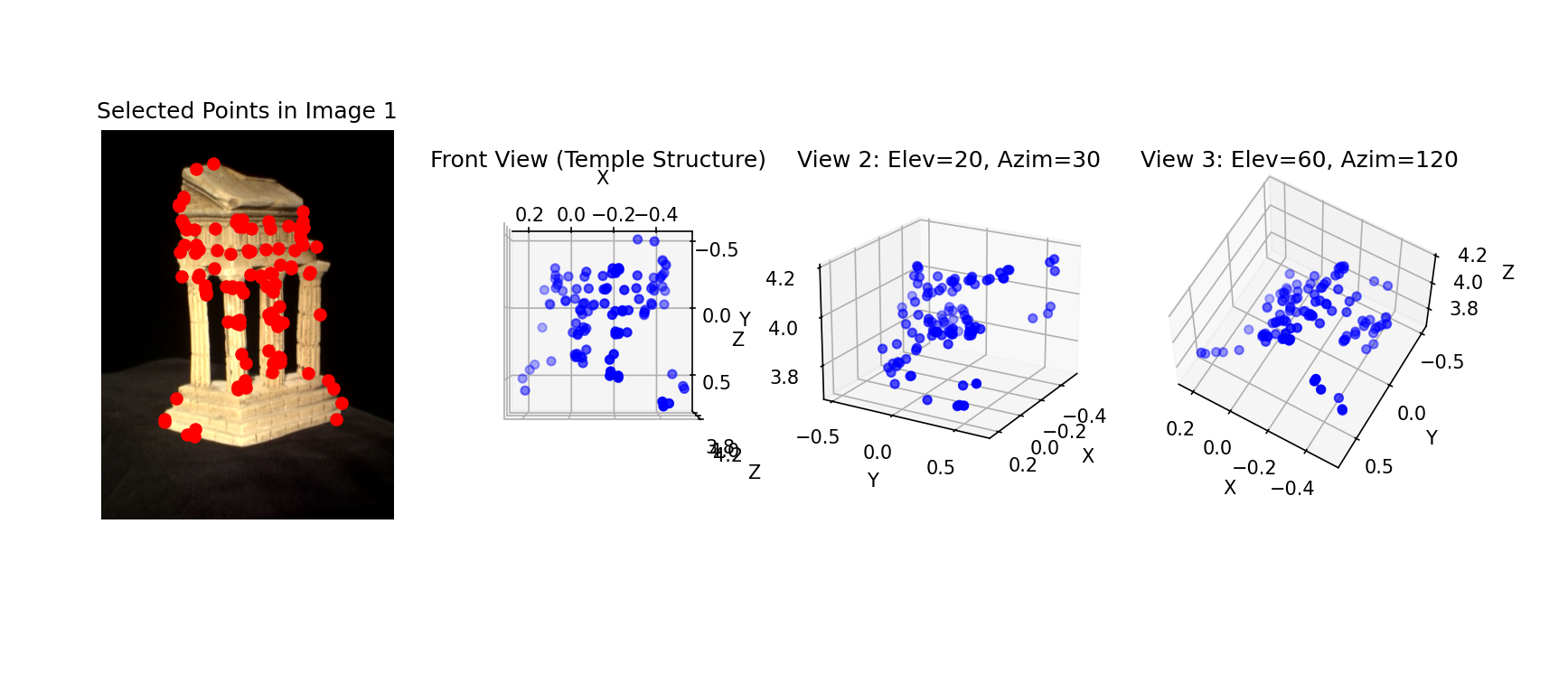
Camera extrinsic is basically the location of the camera in the real world coordinates, comprising of data with both position and orientation. Using the definition of intrinsics(as defined in sec 2.3), we know why the final camera projection matrices is camera intrinsics \* camera extrinsics. We need to find extrinsics of camera 2 to get the final camera projection matrices. We are assuming camera 1 to have no rotation or translation, to be situated at (0,0,0). Camera2 helper function gave 4 possible matrices for extrinsics of P2 from which we needed to get the best P2. The 4 possible solutions that we get are combination of 2 translation values: +ve and -ve of each other, and 2 rotation values, corresponding to 4 possible locations of camera 2 wrt camera 1. To choose the correct solution, we want the position where the 2 cameras are facing the object. For this, we reconstruct 3d from 2d using triangulation method, and then try to find the position of camera 2 where this would satisfy. Reconstruction of 3d from 2d gives errors, hence a simple solution is to consider solution where majority of points are in front of both cameras.



I am getting Reprojection Error as 1.9002, which is under the threshold of 2 pixels.

**2.5**

**Final output Sparse reconstruction**

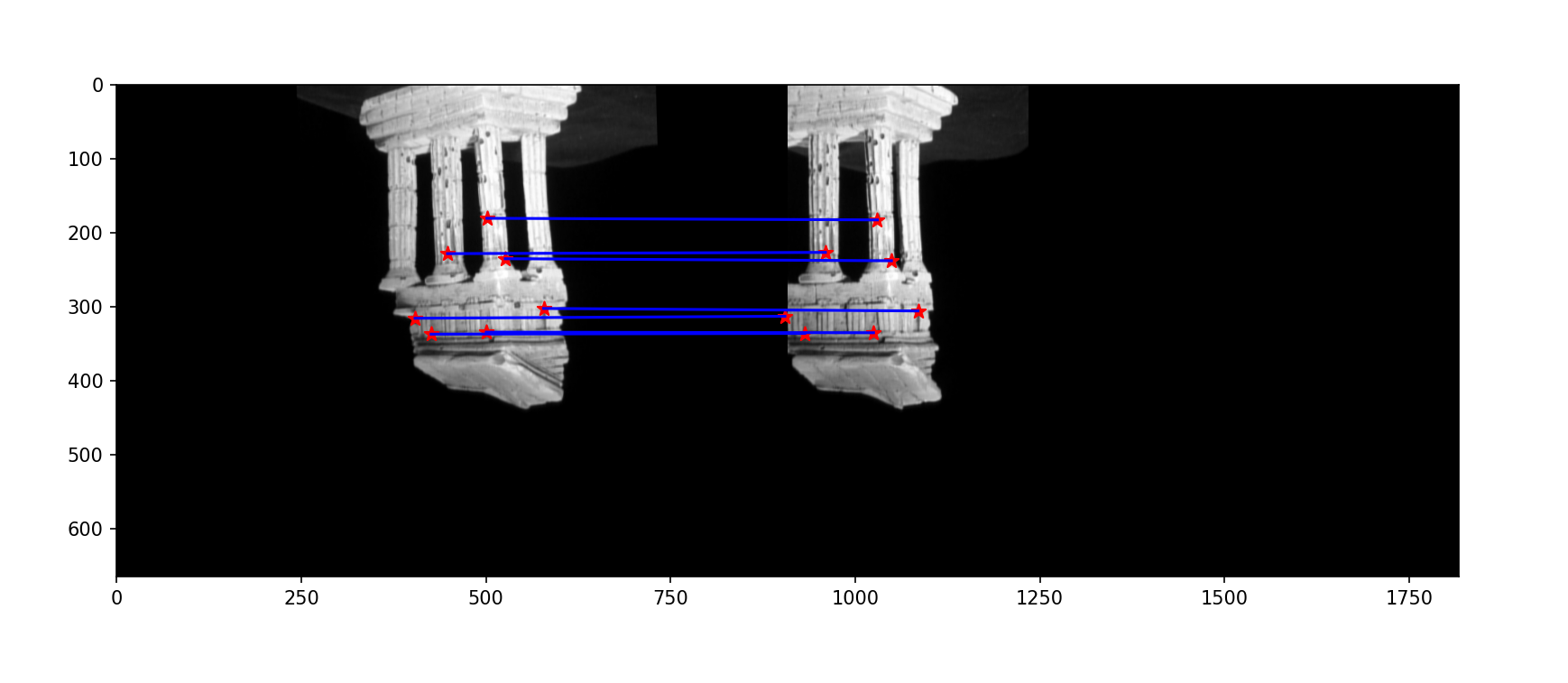
****

Here I have shown the selected points to plot in img1, and the 3d location of these points in 3 different angles, (90, 90), (20,30) and (60,120)

**3.1**

**Image Rectification**

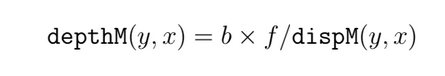
Image rectification is the process of transforming a pair of stereo images so that their epipolar lines become parallel. Makes image planes coplanar, and epipoles at infinity. This simplifies the process of disparity estimation and 3D reconstruction. The rectify\_pair function has been created in the submission file, following the steps mentioned in the problem statement document. Running the test\_rectify function, I got the output below: which has correctly identified the corresponding points in the 2 images and has transformed the images so that the epipolar lines are parallel.



**3.2, and 3.3**

**Disparity map and Depth map**

For 2 images taken with different camera angles, the same point in 3d appears in different positions in the 2 images. The disparity is the horizontal shift between these 2 points. If the images are rectified, it becomes very easy to find the disparity between the images. The farther a point is from the camera, the lesser is the horizontal shift, hence we can say the depth of a point is inversely related to the disparity of that point. Hence depth and disparity are related by the relation:



Where b is the distance between the 2 cameras, and f is the focal length. In the disparity image we got as output, the points which are whiter have more disparity, or have moved more in the images, whereas the points which are darker have moved less. Inversely, for depth image, the points nearer to us are darker, and farther are lighter. We see the output to be somewhat accurate using this algorithm.

