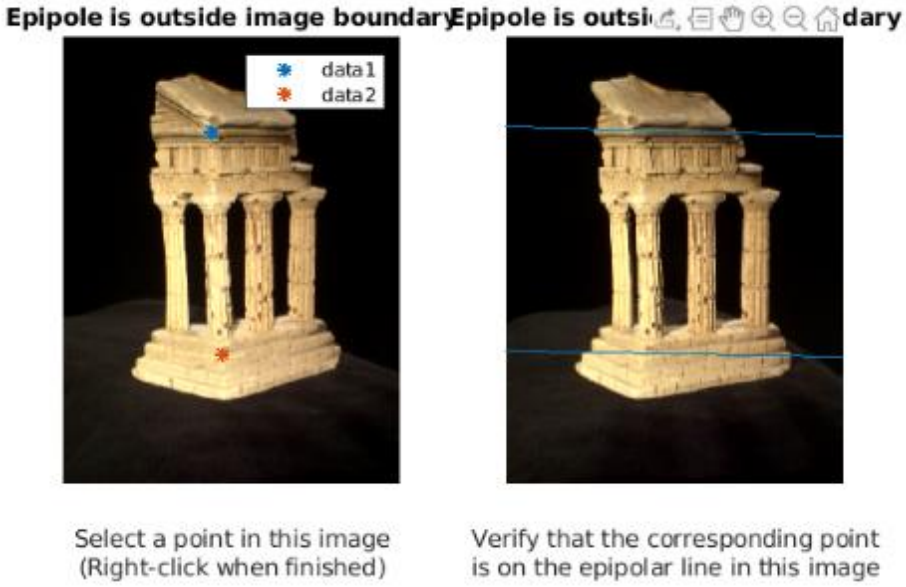
Seerat Kaur Randhawa

301376916

HW3 – GRACE DAY 1(2nd grace day so far)

3.1.1 Implement the eight point algorithm

The following is result from normalized eight point algorithm used to estimate the fundamental matrix to estimate relative correspondence between two camera positions using all the matched points.



The recovered F is :-

-0.0000	0.0000	-0.0000
0.0000	-0.000	-0.0011
-0.0000	0.0010	0.0044

3.1.2 Epipolar correspondences

One problem is if a point is very close to the edge of the image and selecting the window is a problem.

Also, while calculating error using Euclidean distance, due to similarly recurring local patterns, such as on the pillars, the corresponding point is sometimes found on another pillar on the epipolar line. This could also partially because of rotation between the images changes the local pattern. Corners or edges are somehow easier to detect.

The resolution for selecting candidate points can also greatly affect the corresponding match. Too low a resolution can lead to inaccurate matching. Initially tried splitting the possible x coordinate candidates along the epipolar line but rounding off values for indices can lead to bad results. So instead of using techniques like interpolation for getting pixel coordinates, I included x coordinate candidates around possible window on the epipolar line and corresponding y coordinate was also calculated using the epipolarconstraint.

This was again followed by calculation of error based on euclidean distance between selected windows or patches from the two images around the candidate points. The following is the result obtained from epipolarMatchGUI.m :-

Select a point in this image
(Right-click when finished)

Verify that the corresponding point
is on the epipolar line in this image

3.1.3 Compute essential matrix

The intrinsic matrices for both the cameras are the same. The essential matrix is computed by multiplication of the fundamental matrix with the intrinsic matrices.

The computed essential matrix is :-

-0.0029 0.2460 0.0388

0.1788 -0.0001 -1.6241

0.0052 1.6444 0.0019

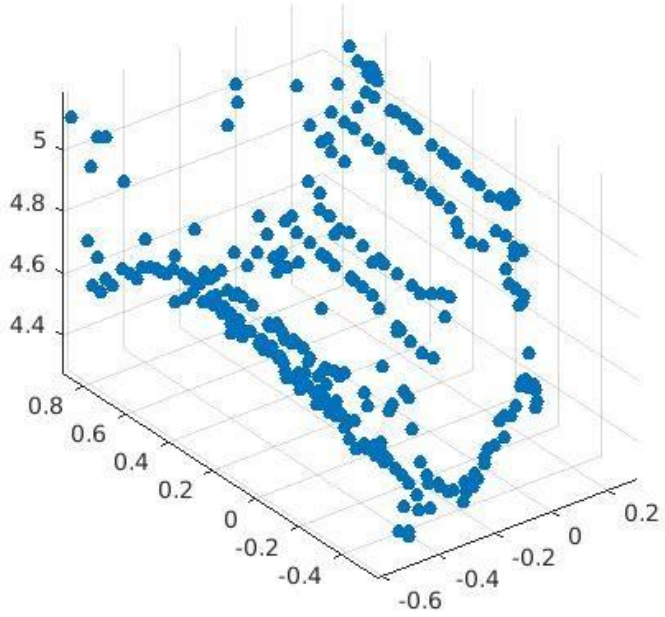
3.1.4 Triangulation

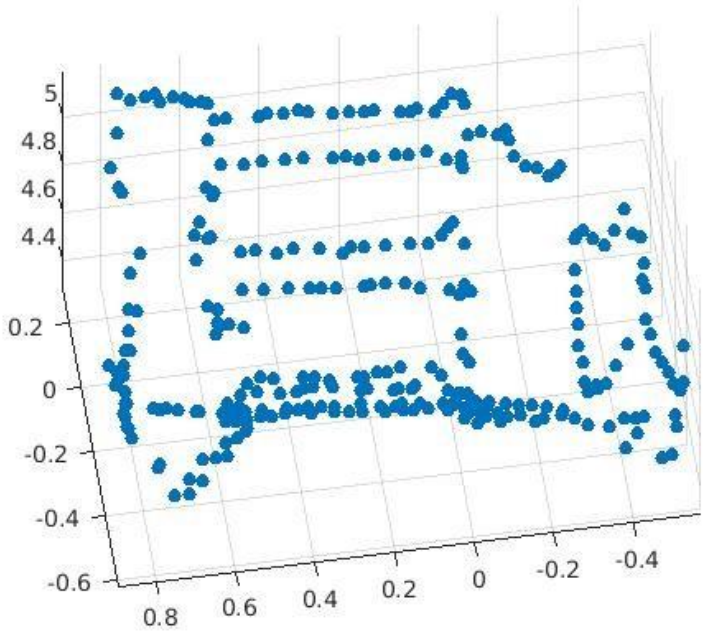
Using the extrinsic matrices, the first with no rotation and translation and the second camera matrix as got from the essential matrix calculated in the previous step, single value decomposition is performed such the constraint x* PX = 0 is satisfied where x is 2d homogenous coordinate and X is the 3d coordinate projected as X on the image. The constraint holds true for projections on both images and hence the 3d points are calculated. The reprojection error obtained by multiplying with the camera matrix to obtain the 2d points back and is not more than one pixel for almost all the point correspondences. The mean error reported over each axis is **1.0868.** The best camera matrix for the second camera is chosen on the basis of positive depth by checking positive z coordinate of the 3d points. For another camera matrix, the reprojection looked like the temple but had all negative coordinates.

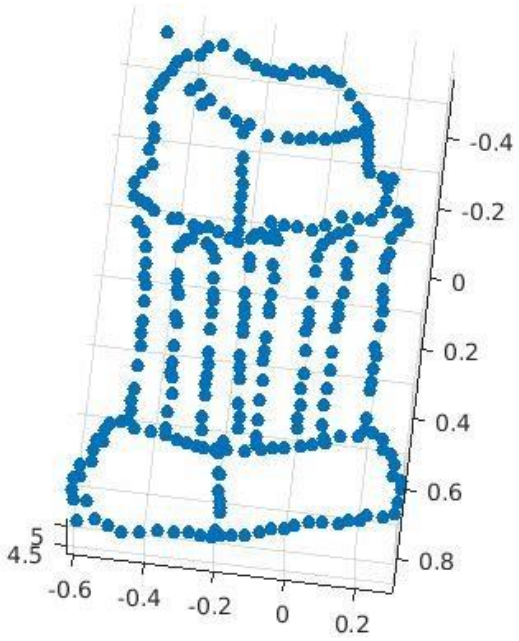
3.1.5 Test script

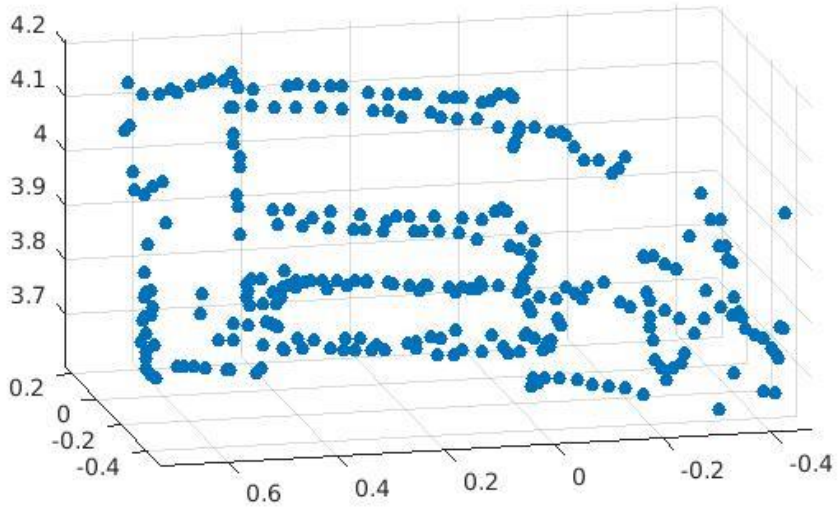
This script is written to load points from temple coords in first image and the corresponding points are found in the second image by calling the epipolar correspondences function. There is one extrinsic matrix for the first camera but four possible candidates for the second camera, out of which the best is chosen on the basis of positive depth by checking positive z coordinate. The extrinsic matrices are decomposed to rotation and translation matrices and the results are saved as mat variables.

The following plots are obtained at different angles of viewing the point cloud.




5
4.8
4.6
4.4
0.2
0
-0.2
-0.4
-0.6
0.8
0.6
0.4
0.2
0
-0.2
-0.4

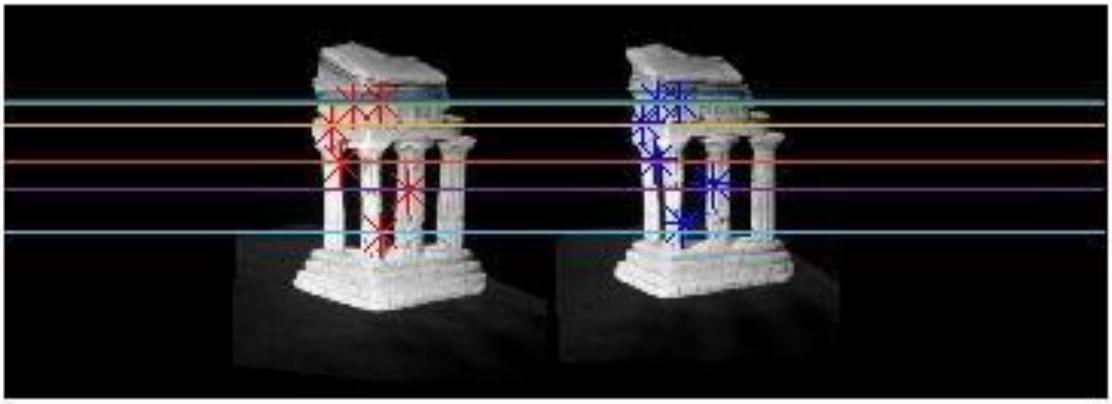

-0.4
-0.2
0
0.2
0.4
0.6
0.8
5
4.5
-0.6
-0.4
-0.2
0
0.2



DENSE RECONSTRUCTION

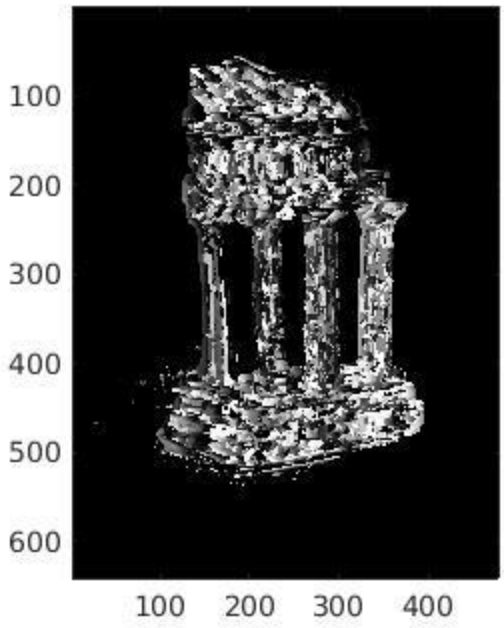
3.2.1 Image rectification

Rectification matrices for both the cameras are calculated in the function rectify_pair so as to project the two images onto the same image plane. For this new artificial camera centers are also computed using the extrinsics rotation and translation saved from the last step. The function is called from test rectify script and the following output is obtained :-



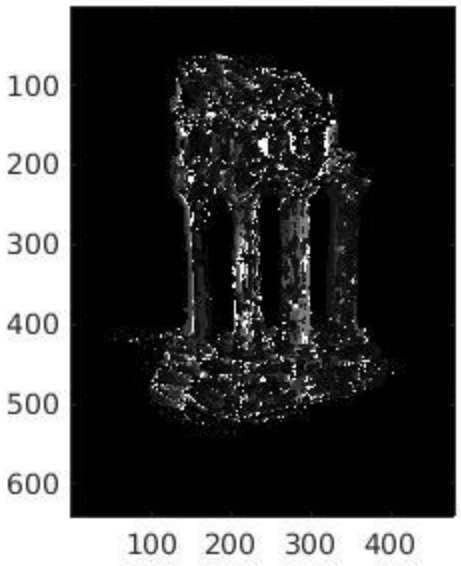
3.2.2 Dense window matching

Dense window matching is carried out for every point in first image along the y axis upto the maximum disparity value. The disparity for every pixel is stored in the disparity map. The calculation of similarity when done solely with squared error leads worse result than absolute error. So squared error was calculated after subtracting the mean from the points and dividing them by standard deviation to make the metric equivalent to cross normalization. Then squared error was calculated. Results such as below are obtained :-



3.2.3 Depth map

Depth map is computed from the disparity map as computed in the previous step by using the formula depth = b × f /disparity. Code was added to rectify images in the testdepth script. The images are rectified and cropped such that the temple is in the centre of the images. The depth map as below is obtained :-



3.3.1 Estimate pose

The camera matrix is solved by solving a system of linear equations that is formed by multiplication of potential camera matrix with 3d points to obtain their 2d image projections. Since, the point correspondences are available, solution for camera matrix is obtained. The output of the script test pose is as follows :-

Reprojected Error with clean 2D points is 0.0000
Pose Error with clean 2D points is 0.0000

Reprojected Error with noisy 2D points is 2.9702
Pose Error with noisy 2D points is 0.1143

3.3.2 Estimate extrinsic/extrinsic parameters

This function was implemented using the algorithm provided to use the inbuilt matlab qr decomposition function for obtaining RQ decomposition. The following is the output obtained from the script :-

Intrinsic Error with clean 2D points is 2.1372
Rotation Error with clean 2D points is 2.0000
Translation Error with clean 2D points is 0.3411

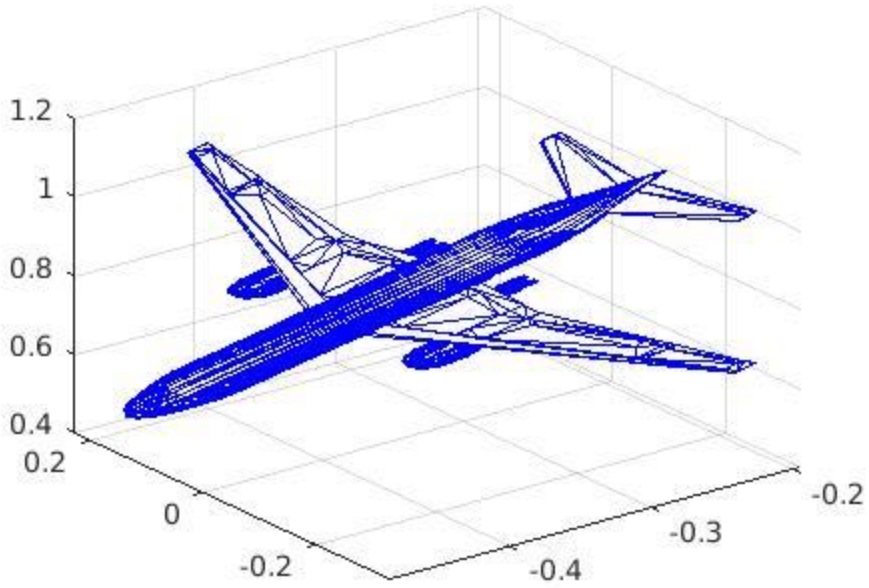
Intrinsic Error with clean 2D points is 1.9395
Rotation Error with clean 2D points is 2.0000
Translation Error with clean 2D points is 0.5334

3.3.3 Projecting CAD model to the image

In the script projectCAD.m, using the functions estimate params and estimate pose from the two previous steps, intrinsic and extrinsics are obtained for the camera. The 3d points are projected back onto the plane image by multiplying with the camera matrix.

The following output is obtained, 3d points projected back are green filled circles and the original 2d points are black circles :-

The rotated CAD model is as shown below :-



Also showing the projected 2d CAD model over the image, in red, below :-