3D Computer vision - PA2

Seung-eun Lee
UNIST
Computer Science and Engineering
selee@unist.ac.kr

2. Sparse Reconstruction

2.1. Implement the eight point algorithm (10 points)

In your write-up: Please include your recovered F and the visualization of some epipolar lines (similar to Figure 3).

Fundamental matrix=

$$\begin{bmatrix} -1.12952541e{-}09 & 1.23285590e{-}07 & -6.23767912e{-}06 \\ 6.40796421e{-}08 & 8.46009618e{-}11 & -1.11138278e{-}03 \\ -1.31648494e{-}05 & 1.06852629e{-}03 & 4.47460714e{-}03 \end{bmatrix}$$

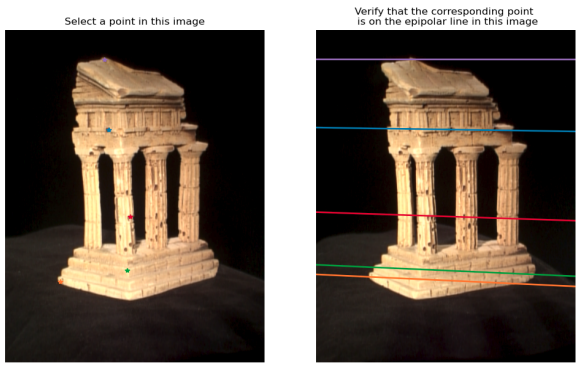

Figure 1. Epipolar lines visualization from displayEpipolarF

2.2 Find epipolar correspondences (20 points)

In your write-up: Please include a screenshot of epipolarMatchGUI running with your implementation of epipolar correspondences (similar to Figure 5). Mention the similarity metric you decided to use. Also comment on any cases where your matching algorithm consistently fails, and why you might think this is.

how to compute similarity for best macth point:

To find the best match point, we select a small window of size w around the point x. Then, compare the target window to the window of the candidate point in the second image. We compute the Euclidian distance and use the Gaussian filter to weight the center point. The case where my matching algorithm consistently fails is that the selected point is on the position where it has similar textures(patches) on the all(or almost) points of the epipolar line. The reason why I think it might fail is that it might be same value to the all of points on the epipolar line if the same pattern is repeated.

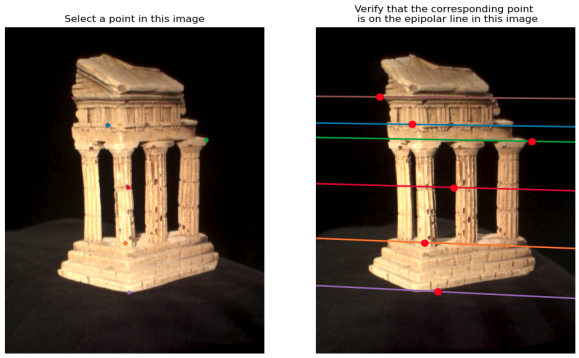

Figure 2. Epipolar Match visualization.

2.3 Write a function to compute the essential matrix (10 points)

In your write-up: Please include your estimated E matrix for the temple image pair.

Essential matrix=

$$\begin{bmatrix} -2.61102920e{-}03 & 2.86019901e{-}01 & 3.62712043e{-}02 \\ 1.48663383e{-}01 & 1.96982410e{-}04 & -1.66626654e{+}00 \\ 3.51670603e{-}03 & 1.68736902e{+}00 & 1.91453960e{-}03 \end{bmatrix}$$

2.4 Implement triangulation (20 points)

In your write-up: Describe how you determined which extrinsic matrix is correct. Note that simply rewording the hint is not enough. Report your re-projection error using the given pts1 and pts2 in data/some corresp.npz.

How I determined which extrinsic matrix is correct is to compute the number of positive depth points(the number of $z \geq 0$). To check that it is correct, I report the re-projection error.

the number of positive depth:

	N_{pos}
$P2_1$	0
$P2_2$	288
$P2_3$	254
$P2_4$	34

Table 1. the number of positive depth

reprojection error:

	pts1	pts2
$P2_1$	2.3861574977541973	2.3386233208417604
$P2_2$	2.3861574977541973	2.3386233208417604
$P2_3$	2.38614340683814	2.3386364343984
$P2_4$	2.38614340683814	2.3386364343984

Table 2. reprojection error

2.5. Write a test script that uses data/templecoords.npz (10points)

In your write-up: Include 3 images of your final reconstruction of the points given in the file data/templecoords.npz, from different angles as shown in Figure 6.

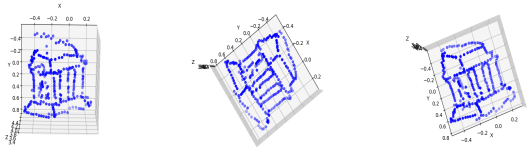

Figure 3. 2.5. 3 images of my final reconstruction of the points from different angles

3.Dense Reconstruction

3.1. Image Rectification (10 points)

In your write-up: Include a screenshot of the result of python/testrectify.py on the temple images. The results should show some epipolar lines that are perfectly horizontal, with corresponding points in both images lying on the same line.

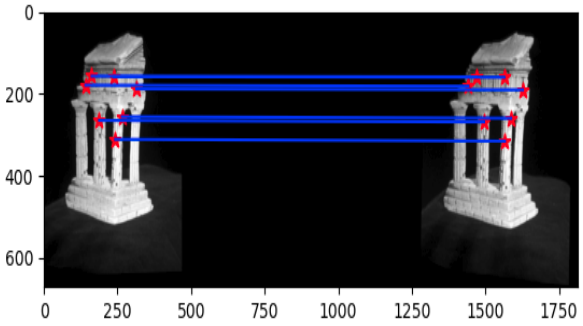

Figure 4. 3.1. Image rectification results

3.2. Dense window matching to find per pixel disparity (20 points) & 3.3. Depth map (10 points)

In your write-up: Please include images of the disparity and depth maps.

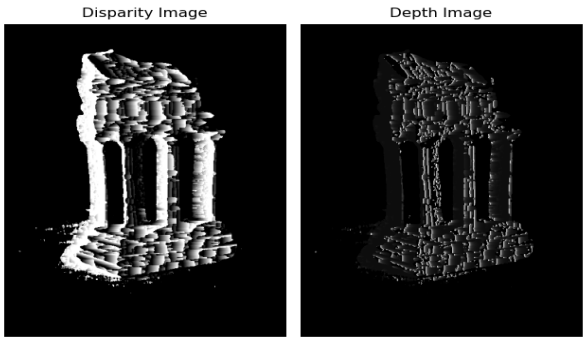

Figure 5. 3.2&3.3 results

4.1 Estimate camera matrix P (10 points)

In your write-up: Please include the output of the script python/testpose.py.

Reprojection Error with clean 2D points	5.89558815580169e-11
Pose Error with clean 2D points	2.5534487850915444e-12
Reprojection Error with noisy 2D points	4.095298564571334
Pose Error with noisy 2D points	2.176135628913067

4.2 Estimate intrinsic/extrinsic parameters (20 points)

In your write-up: Please include the output of the script python/testparams.py.

Intrinsic Error with clean 2D points	2.3746612106544754e-12
Rotation Error with clean 2D points	9.16056365259105e-13
Translation Error with clean 2D points	5.206200033813416
Intrinsic Error with noisy 2D points	0.6696761587081905
Rotation Error with noisy 2D points	0.030345031045376034
Translation Error with noisy 2D points	5.151611534683206

4.3 Project a CAD model to the image (20 points)

In your write-up: Please include the three images similar to Figure 8. You must use different colors from Figure 8. For example, green circle for given 2D points, black points for projected 3D points, blue CAD model, and red projected CAD model overlapping on the image. You will get NO credit if you use the same color.

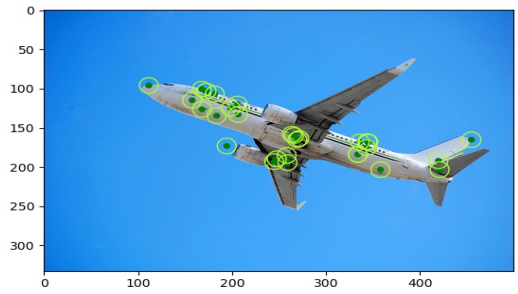

Figure 6. 4.3.4 Plotted image with the given 2D points and the projected 3D points on screen

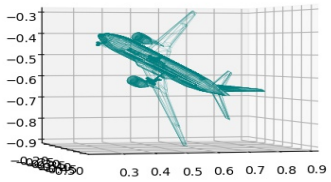

Figure 7. 4.3.5 Drawed image with the CAD model rotated by your estimated rotation on screen

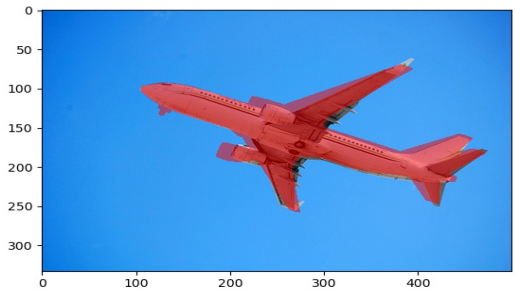

Figure 8. 4.3.6. Image with projecting the CAD's all vertices and drawing the projected CAD model overlapping with the 2D image