3D Reconstruction

CMPT 412 - Computational Vision

Fitzpatrick Laddaran

April 4, 2023

1 Sparse Reconstruction

1.1 Implement the eight point algorithm

This section pertains to Part 3.1.1 of the lab.

Figure 1 visualizes some epipolar lines coded in `eightpoint(pts1, pts2)`. Note that I modified my `eightpoint` function by removing the parameter `M` because when I normalized the coordinates, I instead subtracted the mean and divided the results by the standard deviation of the set (as hinted in the lab).

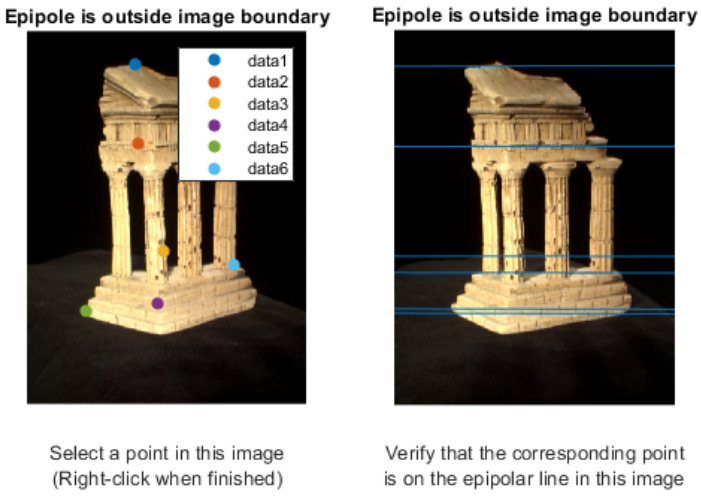

Figure 1: Visualization of some epipolar lines.

The resulting fundamental matrix $\mathbf{F}$ is as follows:

$$\begin{bmatrix} -4.7044 * 10^5 & -1.3390 * 10^7 & 3.9187 * 10^9 \\ -4.9541 * 10^7 & 4.3836 * 10^6 & 1.3875 * 10^{12} \\ -3.4191 * 10^{10} & -1.3863 * 10^{12} & 8.5023 * 10^{10} \end{bmatrix}$$

1.2 Find epipolar correspondences

This section pertains to Part 3.1.2 of the lab.

Figure 2 is a screenshot of `epipolarMatchGui(I1, I2, F)` running with my implementation of `epipolarCorrespondence(im1, im2, F, pts1)`. The similarity metric I used is the Euclidean Distance. My matching algorithm consistently fails in the pillars (within the image). You can see that I selected points on the right-most pillars, which is matched onto the black background. I suspect that this is the case because the image is rotated in a way such that it covers the actual part that I have selected, causing mismatches.

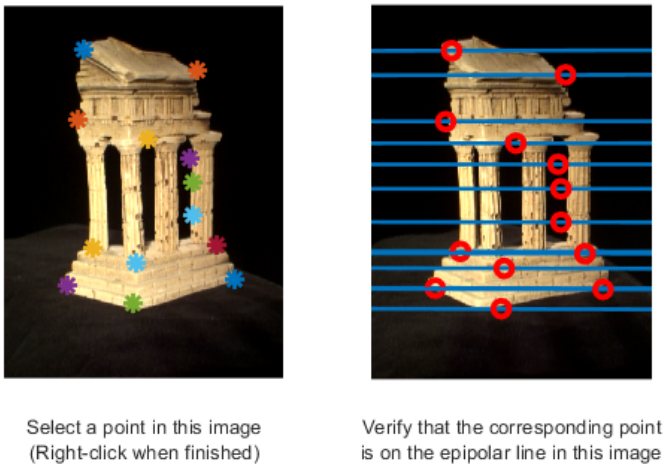

Figure 2: Screenshot of `epipolarMatchGui(I1, I2, F)` running.

1.3 Write a function to compute the essential matrix

This section pertains to Part 3.1.3 of the lab.

The resulting essential matrix **E** is as follows:

$$\begin{bmatrix} -1.0875 * 10^{12} & -3.1064 * 10^{13} & 7.1597 * 10^{11} \\ -1.1493 * 10^{14} & 1.0207 * 10^{13} & 2.0960 * 10^{15} \\ -7.0795 * 10^{13} & -2.1199 * 10^{15} & -1.3242 * 10^{13} \end{bmatrix}$$

1.4 Implement triangulation

This section pertains to Part 3.1.4 of the lab.

I determined the correct extrinsic matrix by determining which candidate had the least number of negative values in the z-axis (as hinted in the lab). Negative values indicate that certain projects are behind the camera, which does not make sense. My re-projection error using `pts1` is 0.2292, and 0.2288 using `pts2`.

1.5 Write a test script that uses templeCoords

This section pertains to Part 3.1.5 of the lab.

Figure 3 is a visualization of my final reconstruction of the `templeCoords` points from one angle. Figure 4 is also a visualization of my final reconstruction of the `templeCoords` points from a different angle, and Figure 5 is another visualization of my final reconstruction of the `templeCoords` points from another different angle.

2 Dense Reconstruction

2.1 Image rectification

This section pertains to Part 3.2.1 of the lab.

Figure 6 illustrates the results of running `testRectify.m`. The results I received are slightly different from what is expected; the horizontal lines exist, but the corresponding points on the right are not exactly (but near) on the line. I followed what was described on the lab, though I do not understand where I could be making a small mistake.

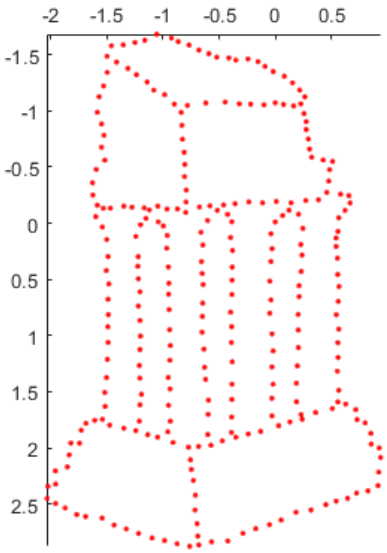

Figure 3: Visualization of final reconstruction of the `templeCoords` points.

2.2 Dense window matching to find per pixel density; Depth Map

This section pertains to Part 3.2.2 and Part 3.2.3 of the lab.

Figure 7 and Figure 8 are visualizations of the disparity and depth map before rectification.

Figure 9 and Figure 10 are visualizations of the disparity and depth map after rectification.

3 Pose Estimation

3.1 Estimate camera matrix P

This section pertains to Part 3.3.1 of the lab.

Figure 11 is the output of the script `testPose`.

3.2 Estimate intrinsic/extrinsic parameters

This section pertains to Part 3.3.2 of the lab.

Figure 12 is the output of the script `testKRt`. I suspect that I made a mistake somewhere in the lab since the errors are quite large; however, I followed the instructions in the lab.

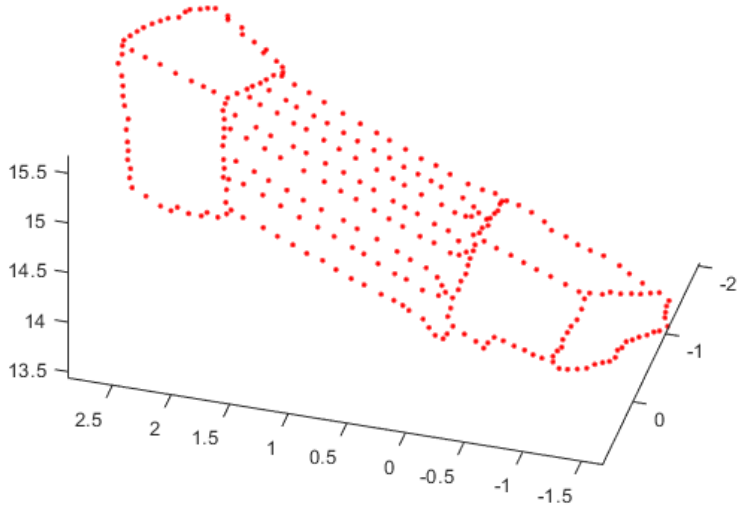

Figure 4: Visualization of final reconstruction of the `templeCoords` points.

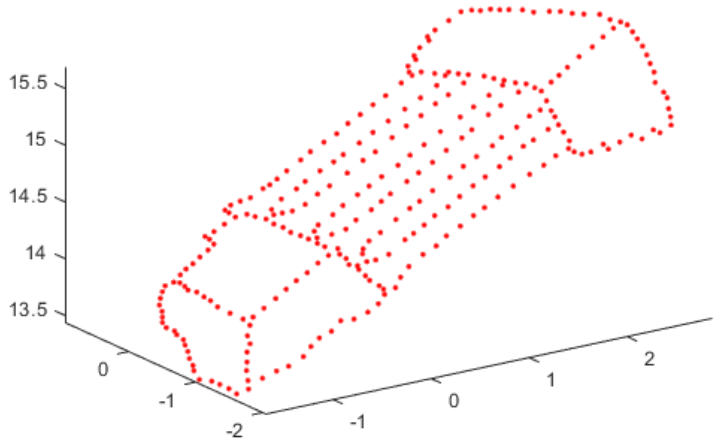

Figure 5: Visualization of final reconstruction of the `templeCoords` points.

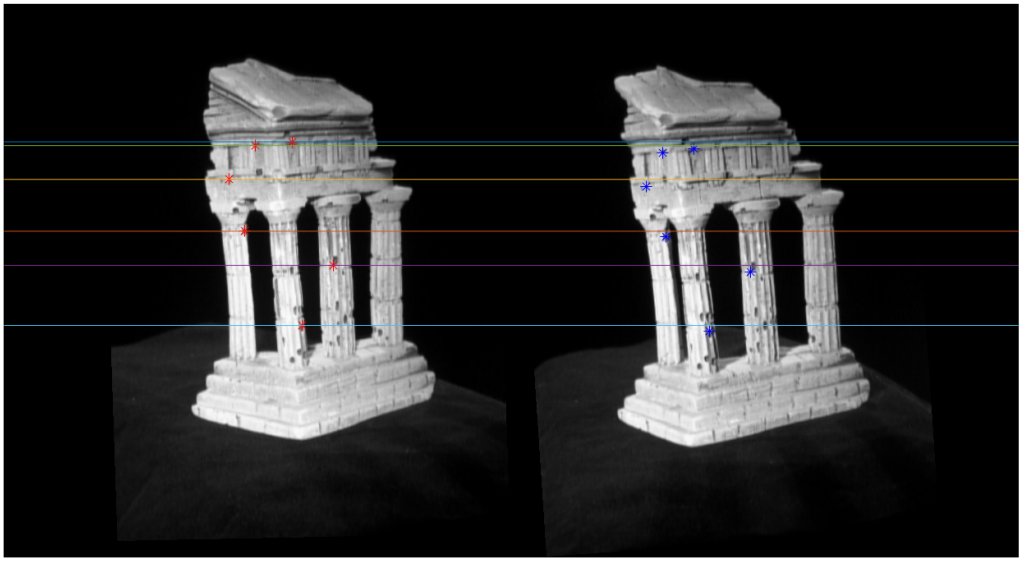

Figure 6: Visualization results of running `testRectify.m`.

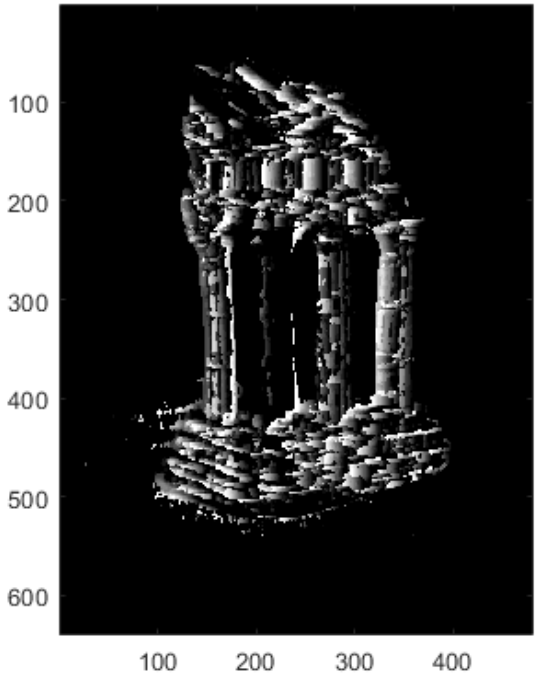

Figure 7: Visualization of the disparity map.

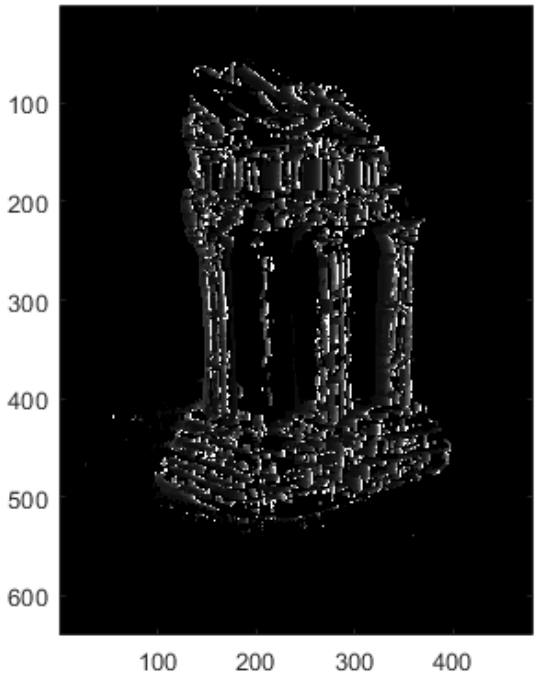

Figure 8: Visualization of the disparity map.

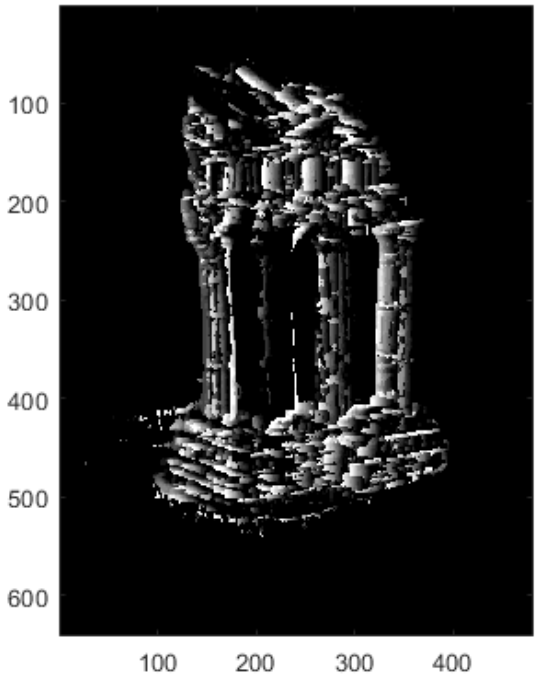

Figure 9: Visualization of the depth map.

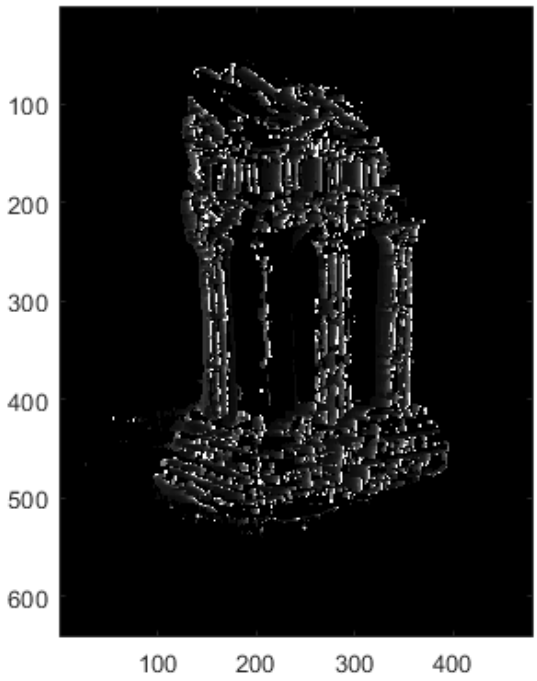

Figure 10: Visualization of the depth map.

```
>> testPose
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 2.9655
Pose Error with noisy 2D points is 0.0139
```

Figure 11: Output of the script `testPose`.

```
>> testKRt
Intrinsic Error with clean 2D points is 140.6002
Rotation Error with clean 2D points is 24371.7110
Translation Error with clean 2D points is 12942.3673
------------------------------
Intrinsic Error with clean 2D points is 140.4514
Rotation Error with clean 2D points is 23873.8288
Translation Error with clean 2D points is 9722.8704
```

Figure 12: Output of the script `testKRt`.