CS 83/183: Assignment 3

February 18, 2024

Jessie Li

2 Sparse Reconstruction

2.1 Eight point algorithm

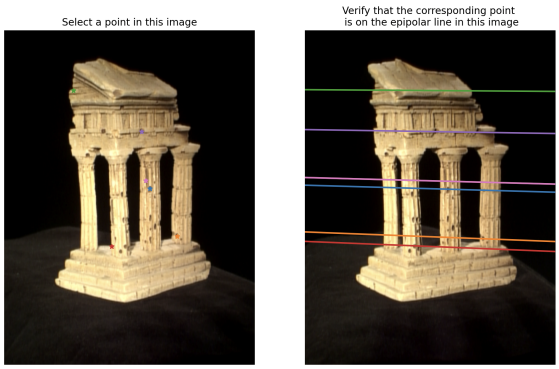


Figure 1: Epipolar lines from `displayEpipolarF`.

2.2 Epipolar correspondences

I chose to use Euclidean distance as my similarity metric.

To match point $\mathbf{x}$ in image 1, I first created an image patch of size 17. For each of the candidate points $\mathbf{x}'$ along the epipolar line in image 2, I created a patch of the same size and calculated the sum of squared differences d (or the square of Euclidean distance). I matched $\mathbf{x}$ from image 1 to whichever $\mathbf{x}'$ in image 2 minimized d.

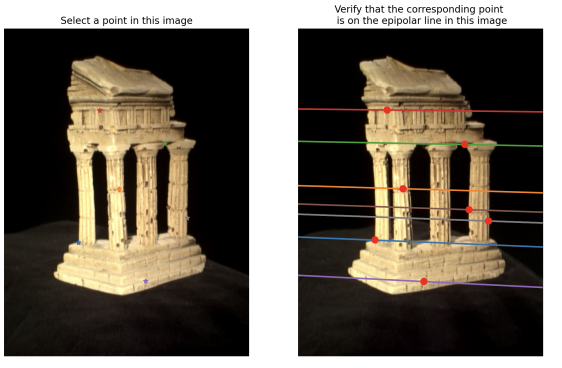


Figure 2: Epipolar matches from `epipolarMatchGUI`.

2.3 Essential matrix

$$\mathbf{E} = \begin{bmatrix} -0.00261 & 0.28599 & 0.03625 \\ 0.14873 & 0.00024 & -1.66626 \\ 0.00353 & 1.68735 & 0.00191 \end{bmatrix}$$

2.4 Triangulation

According to the hint, `camera2` returns 4 candidates for the extrinsic matrix **M2** based on the essential matrix **E**. The correct **P2** "is the one for which most of the 3D points are in front of both cameras." This means that (ideally) all z coordinates of the computed 3D points should be positive, so I picked the **M2** candidate for which `np.all` returned true for all z values in `pts3d`, my $N \times 4$ matrix of (homogeneous) 3D points. Only one of the four matrices satisfied this criteria. If there was no **M2** for which all z coordinates were positive, I would calculate the number of positive z coordinates for each **M2** with `np.sum(pts3d[pts3d > 0])` and pick the **M2** that gives the maximum number of positives.

Mean Euclidean re-projection error for camera 1: 2.193

Mean Euclidean re-projection error for camera 2: 2.149

2.5 3D Reconstruction

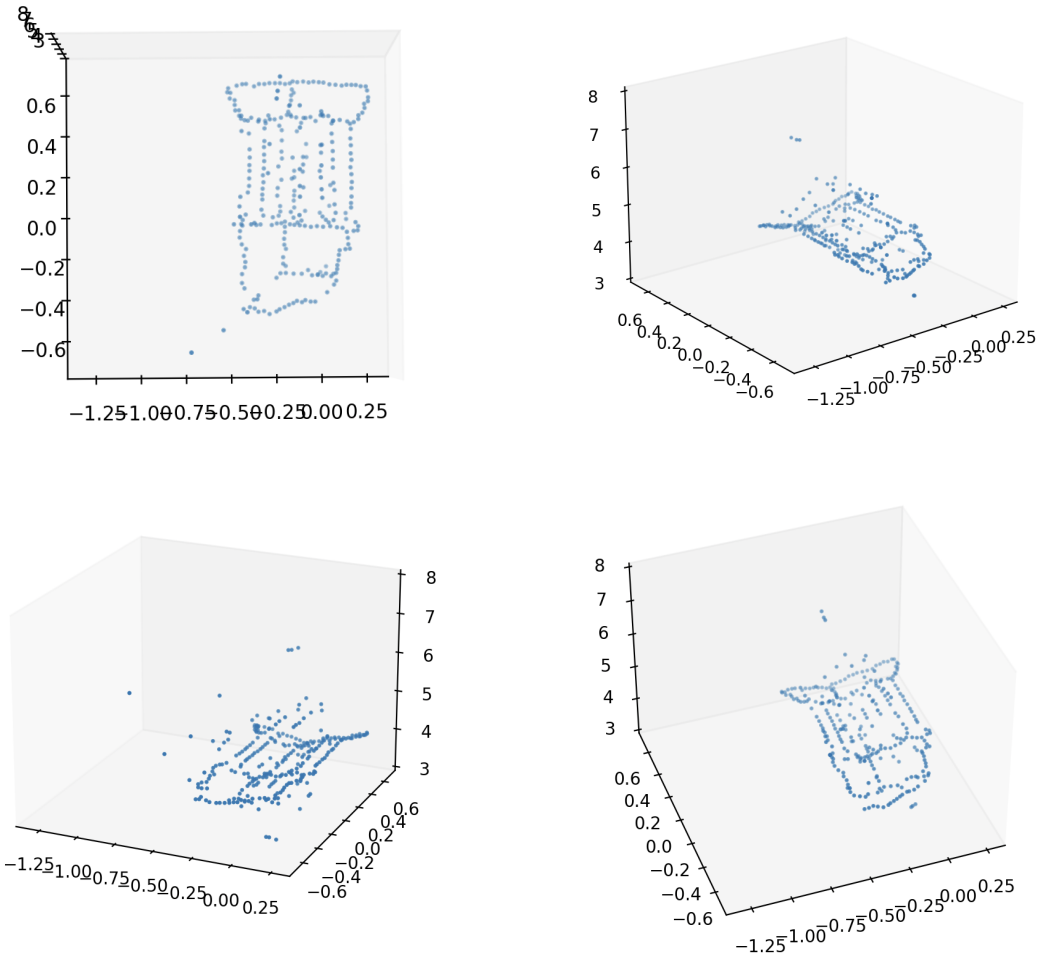


Figure 3: Reconstruction shown from different angles.

3 Dense Reconstruction

3.1 Image Rectification

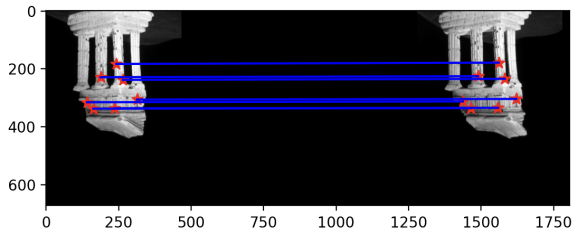


Figure 4: Result of `test_rectify` on the temple images. Epipolar lines are horizontal and corresponding points in both images lying on the same line.

3.3 Depth Map

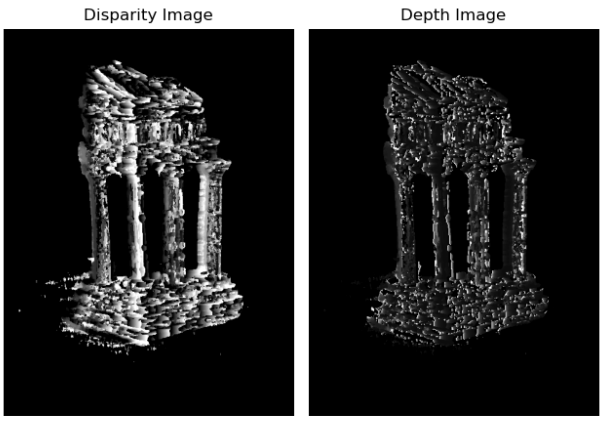


Figure 5: Disparity and depth maps. Brighter pixels indicate higher disparity in the disparity image and are darker (closer) in the depth image.

4 Pose Estimation

4.1 Estimate of Camera Matrix P (Extra Credit)

Output of `test_pose.py`:

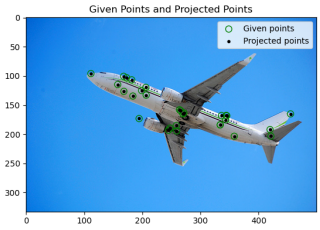
Reprojection Error with clean 2D points:	2.8678476601562916e-12
Pose Error with clean 2D points:	1.9140446602563915e-13
Reprojection Error with noisy 2D points:	5.084786760013155
Pose Error with noisy 2D points:	1.0424290855989087

4.2 Estimate of Intrinsic/Extrinsic Parameters (Extra Credit)

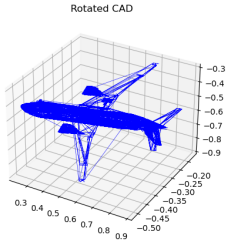
Output of `test_params.py`:

Intrinsic Error with clean 2D points:	1.007573795687616e-14
Rotation Error with clean 2D points:	1.7111094382095604e-14
Translation Error with clean 2D points:	0.34028729560951126
Intrinsic Error with noisy 2D points:	0.7263400732315929
Rotation Error with noisy 2D points:	3.461590328773893
Translation Error with noisy 2D points:	6.132739337160733

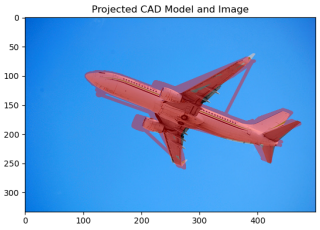
4.3 Projecting a CAD Model to an Image (Extra Credit)



Given points (green circles) and projected points (black points) from the given point correspondences.



CAD rotated by estimated R.



Projected CAD model (all vertices) on top of the image.

Figure 6: CAD projections and rotation based on estimated intrinsic and extrinsic camera parameters.