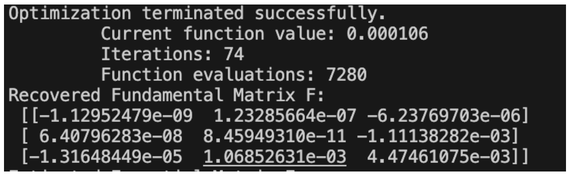
ICS 483, Fall 2025
Programming Assignment 3
3D Reconstruction

Kanta Saito

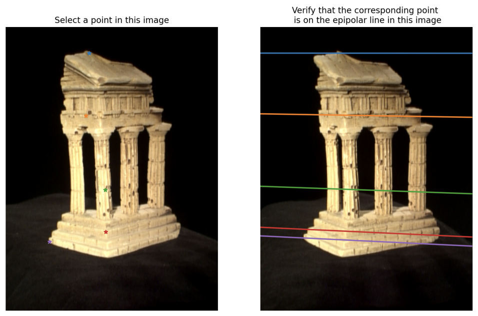
Due Date: (Thur) Nov 13, 2025 23:55 (HST)

Q2.1 Implement the eight point algorithm (10 points)



```
Optimization terminated successfully.
         Current function value: 0.000106
         Iterations: 74
         Function evaluations: 7280
Recovered Fundamental Matrix F:
 [[-1.12952479e-09  1.23285664e-07 -6.23769703e-06]
 [ 6.40796283e-08  8.45949310e-11 -1.11138282e-03]
 [-1.31648449e-05  1.06852631e-03  4.47461075e-03]]
```

(a) Recovered F Matrix


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

(b) Epipolar Lines Visualization

Creating the fundamental matrix, **F**, was simplified by the provided instructions. I had to scale the data by creating a **T** scaling matrix, which I then used to normalize the points. My code also included a check to ensure there were at least 8 points; if not, it raised an error.

Next, I created the **A** matrix as shown in the slides, then used **SVD** to compute **F**. **F** corresponds to the eigenvector associated with the smallest singular value. To enforce the rank-2 constraint (necessary when projecting from 3D to 2D), I used the given **singularize** helper function. Finally, I refined **F** by calling the **refineF** helper function, which singularizes the matrix again to ensure the rank-2 constraint was maintained.

Overall, my resulting image looks similar to the example given in the assignment instructions. From this, I can conclude that the eight-point algorithm works well.

Q2.2 Find epipolar correspondences (20 points)

I used the **ZNCC** (Zero-mean Normalized Cross-Correlation) similarity metric. It is used for template matching and correspondence estimation, acting as a statistical measure that

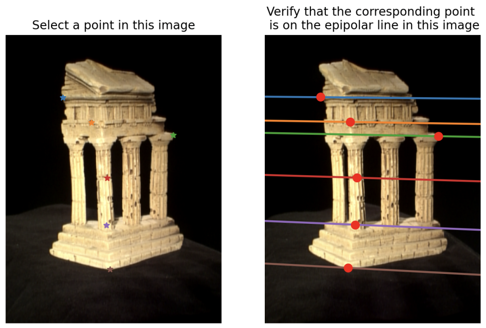
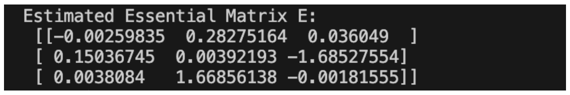

Select a point in this image
Verify that the corresponding point is on the epipolar line in this image

Figure 2: My Epipolar Correspondence

quantifies the similarity between two equal-sized image windows. A high **ZNCC** value (approaching 1) indicates a high degree of similarity and is considered a stronger match. An advantage of **ZNCC** is its **invariance** to affine changes in image intensity, meaning it can compensate for differences in brightness and contrast between the two images.

The cases where the ZNCC similarity metric will fail is when there are geometric transformations/ changes between the template and the target image. It fails to correctly match objects that undergo rotation, scaling, and non-linear deformation.

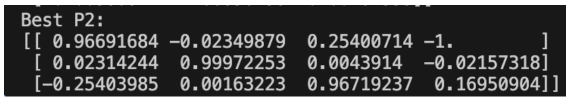
Q2.3 Write a function to compute the essential matrix (10 points)



```
Estimated Essential Matrix E:
 [[-0.00259835  0.28275164  0.036049  ]
 [ 0.15036745  0.00392193 -1.68527554]
 [ 0.0038084   1.66856138 -0.00181555]]
```

Figure 3: My Estimated E matrix

To compute the essential matrix, I used the function: $F = K^{-T}EK^{-1}$ and translated it to: $E = K^{T}FK$. I then made sure we had a rank 2 constraint and also equalized the singular values. I did this by doing **SVD** but did not use the singularize function because of the equalization part.



```
Best P2:
[[ 0.96691684 -0.02349879  0.25400714 -1.        ]
 [ 0.02314244  0.99972253  0.0043914  -0.02157318]
 [-0.25403985  0.00163223  0.96719237  0.16950904]]
```

Figure 4: Best Extrinsic Matrix

Q2.4 Implement triangulation (20 points)

I picked the best extrinsic matrix by applying the cheirality test. This means for each candidate projection matrix $P_2 = K_2[R|t]$, you triangulate 3D points using $P_1 = K_1[I|0]$ and choose the pose that yields the highest count of 3D points with positive depth (i.e., in front of both cameras).This test also ensures a proper rotation by enforcing $\det(R) = +1$. This means if $\det(R) < 0$, both R and t are negated before building P_2 to maintain a consistent projection for the depth checks. If multiple poses are tied for the positive depth count, the one with the smallest mean reprojection error under the corresponding projection matrices is chosen. This method is effective for selecting the physically valid extrinsic matrix from the four potential solutions derived from the essential matrix.

Q2.5 Write a test script that uses data (10 points)

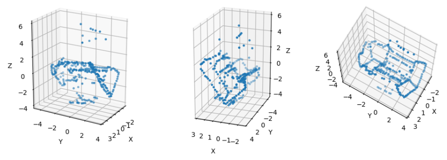
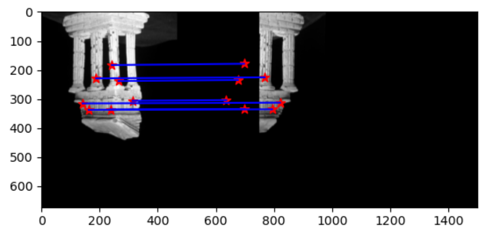
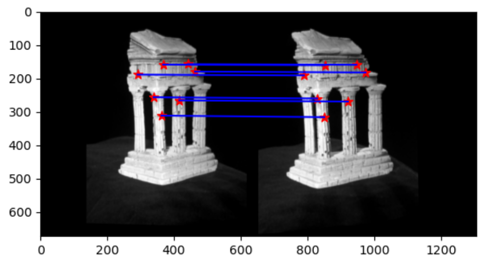


Figure 5: Final Reconstruction Different Points

These three images show my final reconstruction of the given points. To achieve this, I wrote a script that used my temple coordinate data. The most difficult part was performing the triangulation using the projection matrices, because understanding how to check the depth for both cameras was challenging.

Q3.1 Image Rectification (10 points)


(a) Rectify Test (First Test)


(b) Rectify Test (Second Test)

My image rectification does not look very clean. However, I was not given a reference image, so I am unsure if the result is correct. I wrote the rectification function, and when I ran the script, I found that the epipolar lines are completely horizontal. Despite this, the image itself is rendered upside down, and some of the points appear to be detached from the actual image.

I changed the translation of my rectifying matrix, and it finally works as intended. I think the problem was that I had to re-invert the image and then translate it so it was more toward the middle of the image.

Q3.3 Depth map (10 points)

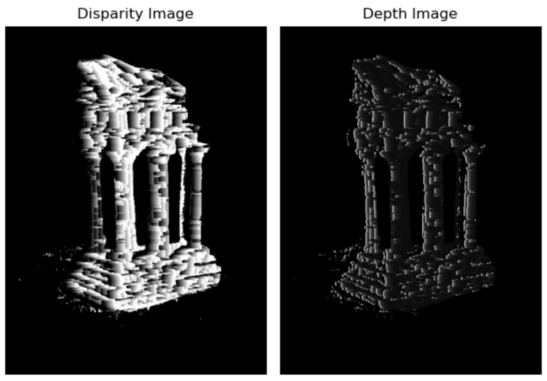
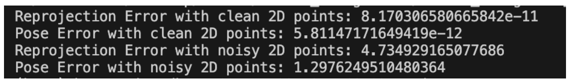


Figure 7: Disparity and Depth Map

These are my disparity and depth maps. Their calculation was the most straightforward. I can tell that these images look good, and I think it's a good result, even though I did not have a reference image. For the disparity map, I used convolution for the cross-term with precomputed sums of squares. For the depth map, I used the given equation along with the baseline and the disparity map.

Q4.1 Estimate camera matrix P (Extra Credits - 10 points)

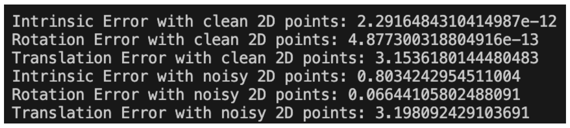


```
Reprojection Error with clean 2D points: 8.170306580665842e-11
Pose Error with clean 2D points: 5.81147171649419e-12
Reprojection Error with noisy 2D points: 4.734929165077686
Pose Error with noisy 2D points: 1.2976249510480364
```

Figure 8: Test Pose Output

The pose estimation shows successful recovery of the camera projection matrix from 2D-to-3D point correspondences. With clean 2D points, both reprojection error and pose error remain extremely small, indicating accurate camera calibration. The performance degradation with noisy 2D points illustrates the sensitivity of the solver to measurement noise and highlights the importance of robust point detection.

Q4.2 Estimate instrinsic/ extrinsic parameters (Extra Credit - 20 points)



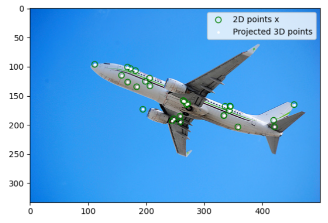
```
Intrinsic Error with clean 2D points: 2.2916484310414987e-12
Rotation Error with clean 2D points: 4.877300318804916e-13
Translation Error with clean 2D points: 3.1536180144480483
Intrinsic Error with noisy 2D points: 0.8034242954511004
Rotation Error with noisy 2D points: 0.06644105802488091
Translation Error with noisy 2D points: 3.198092429103691
```

Figure 9: Estimate intrinsic/extrinsic parameters

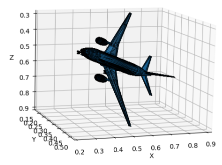
The decomposition of the camera matrix into intrinsic and extrinsic components successfully isolates the camera's internal properties from its external pose. The intrinsic error with clean data is negligible, while rotation and translation errors remain at extremely small values, confirming the mathematical decomposition is correct. The same is true for noisy conditions.

Q4.3 Project a CAD model to the image (Extra Credit - 20 points)

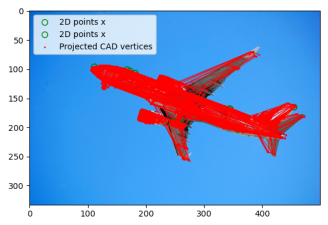
This was successfully replicated to match the given figure. I had a problem displaying the CAD model using Trisurf because I was using regular plot functions.



(a) Image annotated



(b) CAD model rotated by R



(c) Image overlapped with CAD

LLM Usage/ References

0.1 LLM Usage

How do I use Trisurf to plot my CAD model?
Why is my rectified image upside down?
What is ZNCC and how do I implement it?

0.2 References

Wilson, Christian.