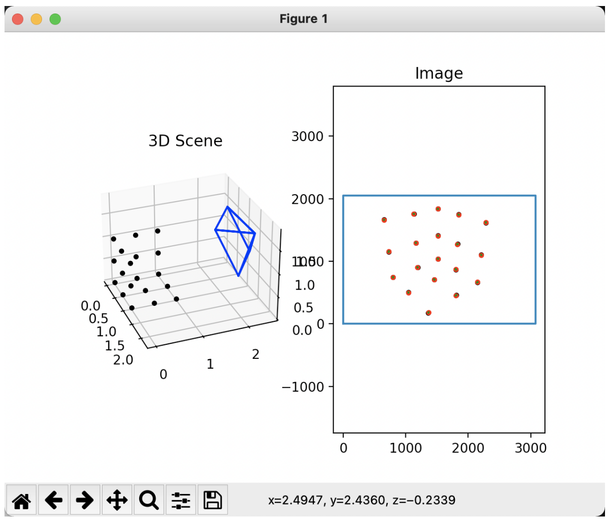
CV Assignment 3, Camera Calibration & SfM

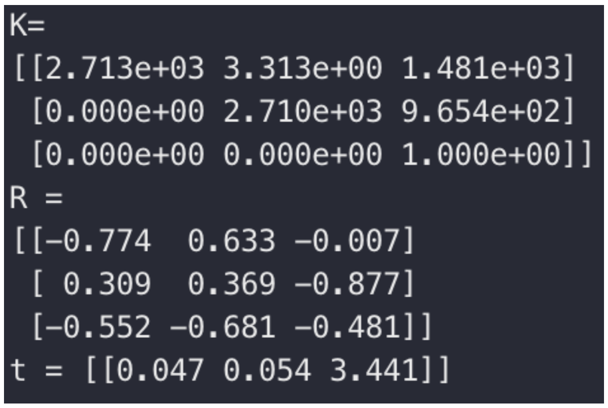
Jingyu Liu

Calibration

Since our implemented model is $P = K[R|t]$ which is a linear model, any nonlinear distortion like radial distortion can not be modeled with our approach.

a) Data Normalization: Just applied the helper function and saved the transformation matrices which then multiply to the results with proper order (taking inverse if need to).
b) DLT: Built the constraint matrix according to the Gold Formula given in the lecture with the 2D points treated as inhomogenous coordinates (or w=1 as homogeneous). Then use SVD to
find the null vector which gets reshaped to 3x4 to get our projection matrix. Each 2d-3d correspondance will create two independent constraints.
c) Optimizing reprojection errors: yes, the optimization step reduces the error from 0.0006316426059796237 to 0.0006253538899291131.
d) Denormalizing the projection matrix: use the equation $P = T_{2d}^{-1}\widehat{P}T_{3d}$ to find the denormalized P.
e) Decomposing the projection matrix: followed the logic from the handout (use QR decomposition and afterwards maintain the constraints for K, R and t). The final reprojection error is 0.0006253538899291131 with the following visualization:





```
K=
[[2.713e+03 3.313e+00 1.481e+03]
 [0.000e+00 2.710e+03 9.654e+02]
 [0.000e+00 0.000e+00 1.000e+00]]
R =
[[-0.774  0.633 -0.007]
 [ 0.309  0.369 -0.877]
 [-0.552 -0.681 -0.481]]
t = [[0.047 0.054 3.441]]
```

SfM

a) Essential matrix estimation: first normalize both image planes and then use the algorithm presented in the slides to solve for the null space. The constraint for each 2d2d correspondance is $[x'x,\ x'y,\ x',\ y'x,\ y'y,\ y',\ x,\ y,\ 1]$ assuming points are normalized homogeneous (w=1). Then SVD decompose the matrix and manually set the first and second singular values to be 1 and the third 0, and then reconstruct the essential matrix with $E = Udiag([1,1,0])V^T$.

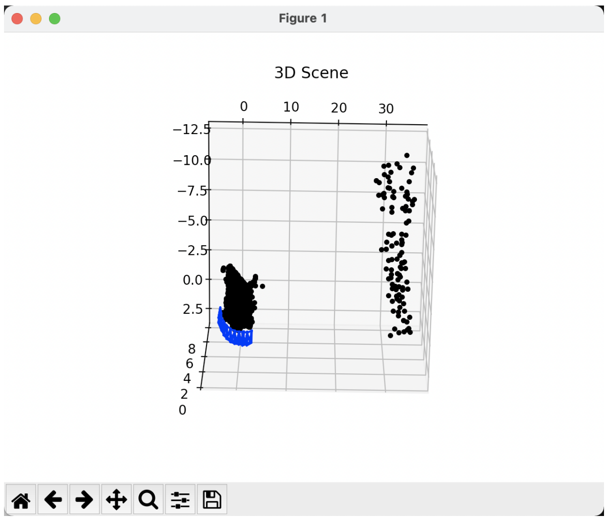
b) Point triangulation: already implemented, just filter the points which, when projected to the image plane, have negative depth.

c) Finding the correct decomposition: due to the assert statement in the essential matrix finding function, we're computing the essential matrix from image 2 to image 1, hence we set the pose for image 2 to be $[I|0]$, and try the four poses for image 1. The final pose is the one with the most triangulated points in front of both cameras. There are two cases that have different results:

1. If we use $[R|t]$ as the pose for image 1 and [I|0] for image 2, we will have the numerical issues, which is discussed below.
2. If we use [I|0] as the pose for image 1 and $\left[R^T| - R^Tt\right]$ for image 2, which takes the inverse of the essential matrix, we will not have the problem of choosing 3, 4 as the initial images.

d) Map extension: Then iteratively add new image to the final results. Each image gets the 3d correspondances with the matches information and the 3d points already computed. Then a standard projection matrix DLT is conducted, which completes the pose estimation. Then we triangulate the new image with all registered images to get new 3d correspondances which will update both the 3d point sets and the correspondances for each registered image. This will continue until all images are registered.

The final result we get using [3, 4] as the initial image 1 and 2 and using the inverse of E to set poses will be:



However, there are some problems:

1) If we just use E as the essential matrix from image 2 to image 1, the final result depends on the choices of the initial two images between which we compute the essential matrix

(using 3 and 4 will produce very large noises, resulting in far away triangulated points. using 5 and 6 is a lot more stable). This problem, however, can be solved if we only keep the triangulated points from the initial two cameras and do not add further 3d points using triangulations (so there might be some error in this procedure that causes the instability). Or we use the inverse of the essential matrix and set the [I|0] as the pose for im1 and $\left[R^T|-R^Tt\right]$ for im2. The following are reproduced when we use the original E.

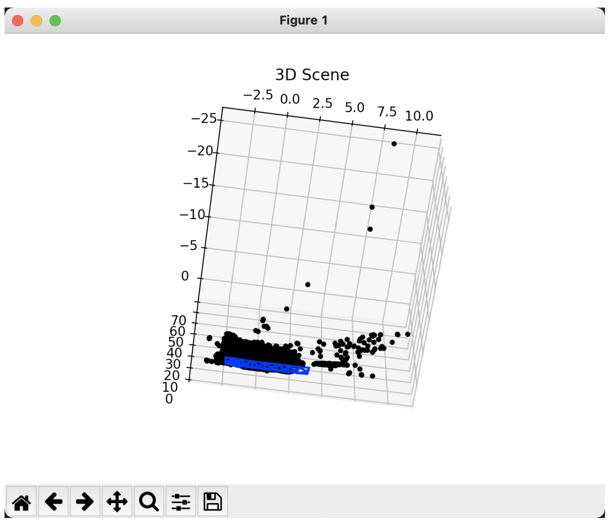


Figure 1: [3,4] initial images with 3D points

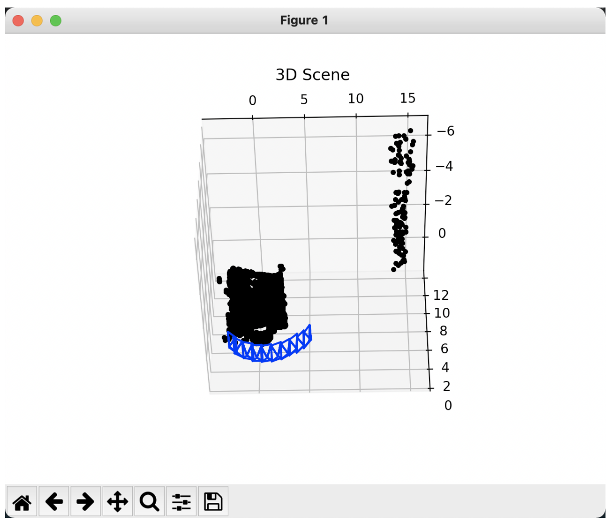
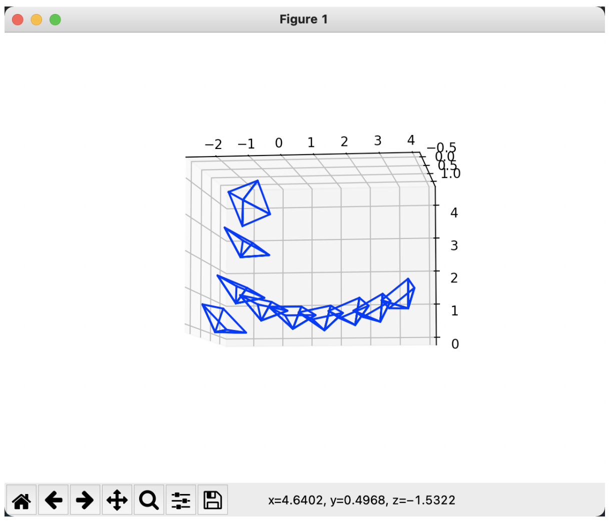


Figure 2: [5,6] initial images with 3D points



x=4.6402, y=0.4968, z=−1.5322

Figure 3: [3,4] initial images with only cameras

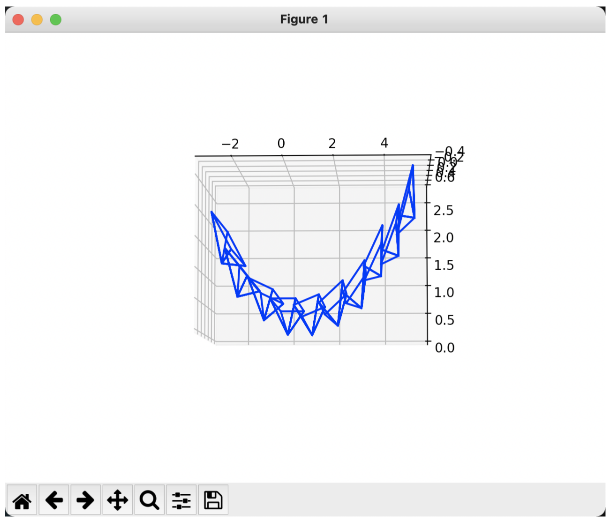


Figure 4: [5,6] initial images with only cameras

2) The plotting graph has different scales for xyz coordinates, making the shape of the camera not natural. This can be easily fixed by manually setting them equal (using the self-implemented function called AxisEqual3D), which produces the result:

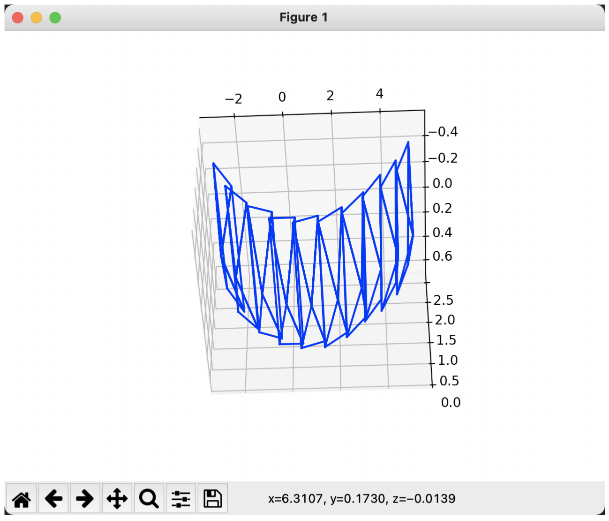


Figure 5: axis with different scales

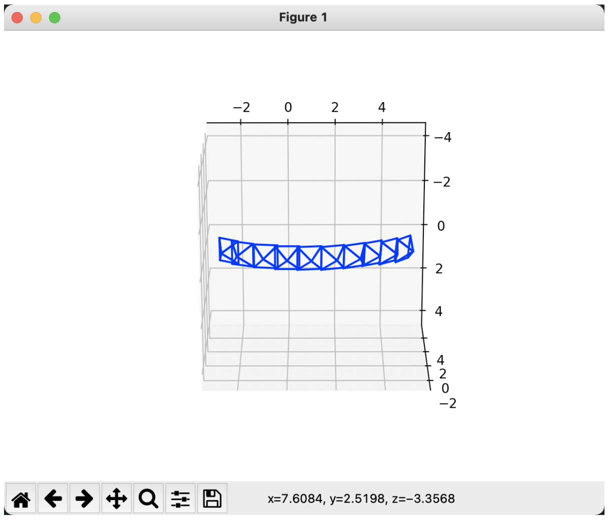


Figure 6: axis with equal scales