Homework 4 - Structure from Motion

Naomi Derel, 325324994 & Sagi Ben Lulu, 207031493

Question 1 - Sparse Reconstruction

1.1 - Eight Point Algorithm

In this section we implemented the 8 Point Algorithm to find the fundamental matrix between 2 images from a set of point matches from the images.

First, we had to normalize the points with matrix T by applying $p = Tp$, after that we created the matrix A as shown in the lecture to solve a linear equation $AF = 0$:

$$\begin{bmatrix} x_1x'_1 & x_1y'_1 & x_1 & y_1x'_1 & y_1y'_1 & y_1 & x'_1 & y'_1 & 1 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ x_Mx'_M & x_My'_M & x_M & y_Mx'_M & y_My'_M & y_M & x'_M & y'_M & 1 \end{bmatrix} \begin{bmatrix} f_1 \\ f_2 \\ f_3 \\ f_4 \\ f_5 \\ f_6 \\ f_7 \\ f_8 \\ f_9 \end{bmatrix} = \mathbf{0}$$

Solved it with SVD to find F and then using SVD again we forced the rank 2 constraint on F, then refined F using the provided helper function, and finally we unnormalized by applying $F = T^T F T$, and got this fundamental matrix:

```
Fundermental matrix:
[[-1.12821729e-09  1.23273596e-07 -6.24808684e-06]
 [ 6.41083704e-08  1.04710755e-10 -1.11138905e-03]
 [-1.31615504e-05  1.06851393e-03  4.47845837e-03]]
```

Given the matrix F the epipolar line corresponding to a point is calculated by $l' = Fp$. The matrix F is calculated with SVD over a lot of points so it is more robust and has a good fit to all points therefore the epipolar lines should get a good match to the point.

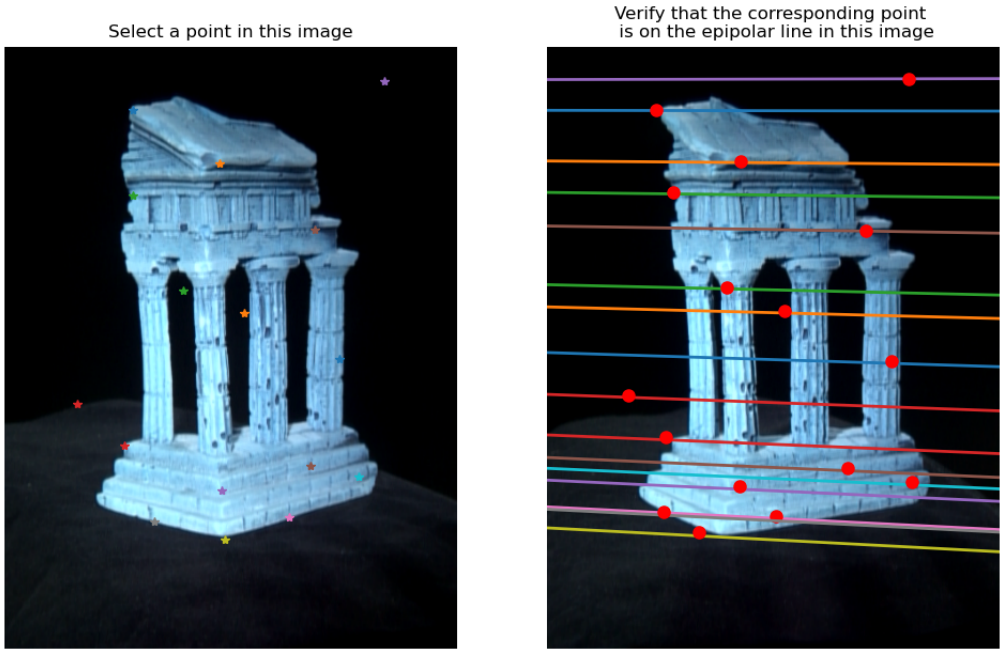
Results from the display_epipoles function show for each point chosen in the left image the corresponding epipolar line in the right image, which seem correct:

1.2 - Epipolar Correspondence

In this section we tried to match a point in one image to a point in the other. First we found the corresponding epipolar line in the other image, then found candidates for the matching point by defining the x range across the whole image and computing the corresponding y values on the Epipolar line, within bounds.

To find the best matching, we padded the image (with zeros) and took a window around each candidate point, comparing it to a window around the original point in image 1. We calculated the Euclidean distance between the windows and kept the candidate with the minimal distance as out matching. We empirically experimented and chose the patch size to be 64, to reach the desired low error in the next steps.

Our algorithm preforms well on most of the points, as can be seen in the results that show the corresponding points:



We found that some points on the left side of the temple, which has limited visibility in the second image, preform worse than others – the green point is wrongly found on the pillar instead of between them, and the pink point at the base is shifted. This is likely due to dissimilar windows around those pixels due to the rotation of the image.

1.3 – Essential Matrix

The Essential Matrix is calculated by the given intrinsic camera matrices k1, k2 with the equation - $E = K2^T F K1$, and the result is:

```
Essential matrix:
[[-2.60800532e-03  2.85992074e-01  3.62514796e-02]
 [ 1.48730032e-01  2.43805465e-04 -1.66625527e+00]
 [ 3.53310602e-03  1.68735221e+00  1.91423913e-03]]
```

1.4 – Triangulation

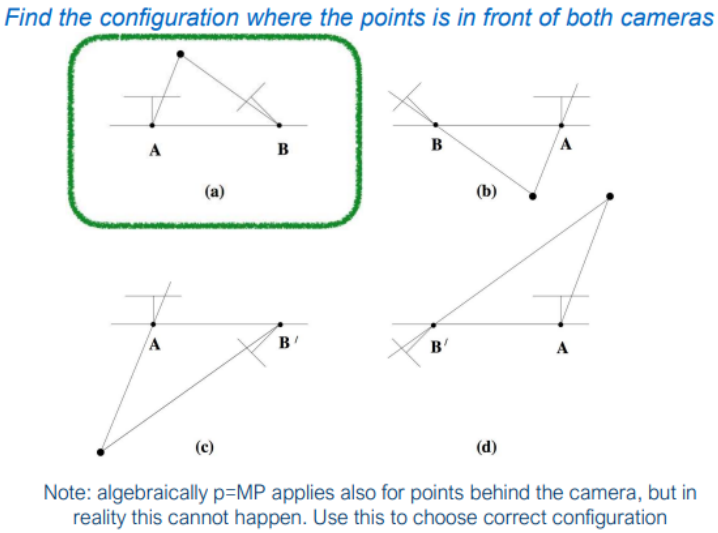
We can triangulate each pair of points p1, p2 by constructing the matrix:

$$\begin{bmatrix} ym_3^T - m_2^T \\ m_1^T - xm_3^T \\ y'm'^T_3 - m'^T_2 \\ m'^T_1 - x'm'^T_3 \end{bmatrix} P = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

And solving for P using SVD.

The helper function provided returns the 4 options for camera matrix M2 given the essential matrix. To find the correct one , we used each of them to triangulate the points matches we previously computed, and chose the matrix M2 that got the most 3D points in front of the cameras (the Z value is greater than zero).

For each M2 matrix we held a score and increased it for every 3D point in front of each camera, and the final scores are [0, 576, 288, 288], which aligns with our intuition that 1 matrix will have the point in front both of the cameras, 2 matrices will have only 1 point in front of the camera, and 1 matrix will have both points behind it.



After finding the correct M2 matrix, we used it to triangulate all the point matches and find their corresponding 3D point. We then project the points back to each image, and compute the re-projection error by computing the mean Euclidean difference between the original and reprojected points. Our results are:

The error in image 1 is: 1.489
The error in image 2 is: 1.457

Which are lower than 2, as instructed.

1.5 – Putting It All Together

In this section we combined all the previous functions to get the 3D reconstruction.

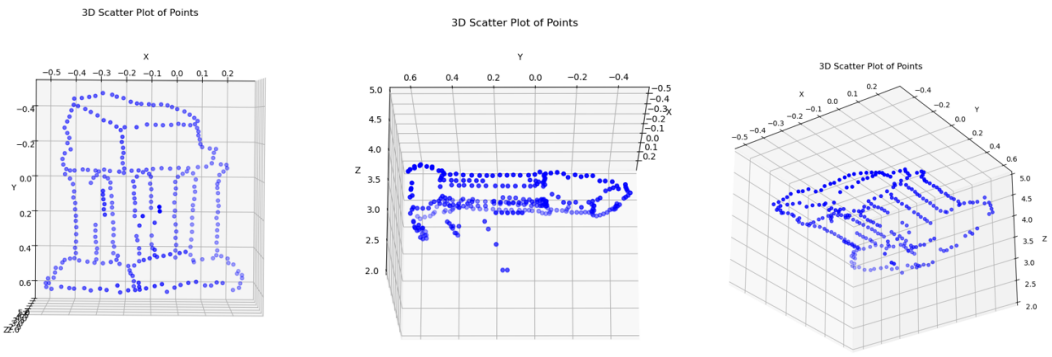
The selected matrix M2 is:

```
Camera matrix 2:
[[ 1.39328123e+03 -3.52384868e+01  6.78630399e+02 -1.46915835e+03]
 [-2.74125748e+01  1.52587647e+03  2.45489654e+02  8.94241679e+00]
 [-2.54065389e-01  1.61961591e-03  9.67185688e-01  1.69494744e-01]]
```

And the extrinsic parameters are:

```
Extrinsic parameters for camera 1:
R1 =
[[1. 0. 0.]
 [0. 1. 0.]
 [0. 0. 1.]]
t1 =
[0. 0. 0.]
Extrinsic parameters for camera 2:
R2 =
[[ 0.9669102  -0.02349916  0.25403237]
 [ 0.02313949  0.99972254  0.0044043 ]
 [-0.25406539  0.00161962  0.96718569]]
t2 =
[-1.         -0.02156154  0.16949474]
```

The result of the 3D reconstruction is:



We can see the shape on the temple in the picture is well preserved, and the lines in the pillars are clear and straight. The depth is logical, with the front pillar and platform being more outward in a triangular shape, as we see in the image.

However, there are a few noisy outliers that are projected way closer than they should be, and cause disruptions in some of the pillar lines.

Question 2 – Pose Estimation

2.1 – Estimating M

We compute M by defining a matrix A from pairs of points p and P, as in the lecture, then computing the SVD and reshaping the smallest eigenvector to the matrix M of 3x4.

The resulting errors are:

```
Reprojection Error with clean 2D points: 2.1689452576250208e-10
Pose Error with clean 2D points: 5.609536336215084e-12
Reprojection Error with noisy 2D points: 6.74234962730379
Pose Error with noisy 2D points: 1.2968141360053482
```

2.2 - Intrinsic/Extrinsic Parameters

We estimate the parameters from the matrix M:

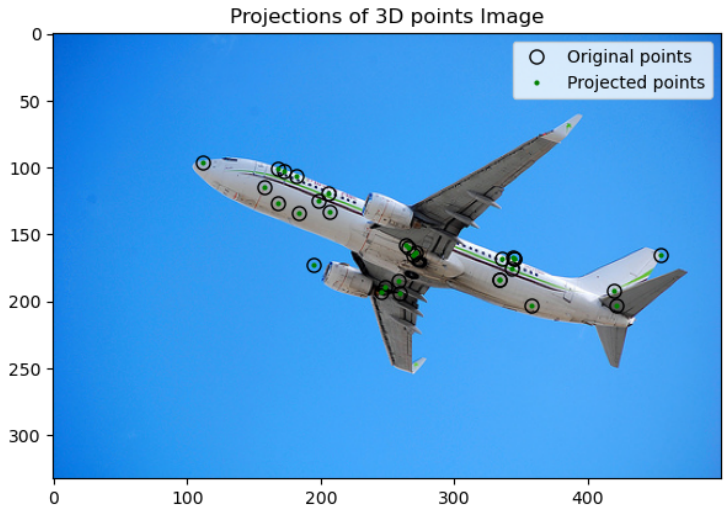
- The R and K matrix as the QR-decomposition of M respectively, where the decomposition is computed on the inverse of M and the results are inversed once more to match the requirements of the matrices (upper triangular and orthogonal).
- The transition by computing $t = -Rc$, where the camera center is the smallest eigenvector in the SVD decomposition of M.

The resulting errors are:

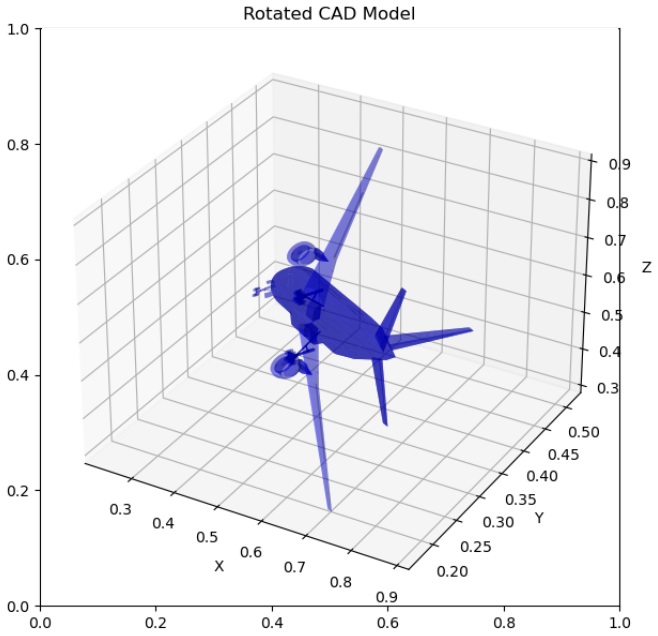
```
Intrinsic Error with clean 2D points: 3.2624018536812572e-12
Rotation Error with clean 2D points: 1.5693500729910895e-12
Translation Error with clean 2D points: 3.3896555323272803
Intrinsic Error with noisy 2D points: 0.8131507530494083
Rotation Error with noisy 2D points: 0.016135987769038317
Translation Error with noisy 2D points: 3.4808182412450037
```

2.3 – Projecting CAD Models to 2D

In our first result, we project the given 3D points back to the image, using the computed matrix. The plot shows the original and projected points:



In our second result, we rotate the CAD model using our estimated matrix R and applying it on the CAD vertices. The resulting model in 3D:



In our third result, we use the matrix M to project the CAD model, the vertices, onto our image. We show them overlapping: