Calibration and Sturcture from motion

Qi Ma, 20-960-225, qimaqi@student.ethz.ch

November 18, 2021

Abstract

This assignment teaches me use basic knowledge about image formation and multi-view camera and solve correspondent questions.

1 Calibration

The first task is about calibrating camera using checkerboard. The most important part is solving DLT problem.

Q1) Given the camera models that you saw in the lecture, which part of the full model is not calibrated with our approach?

A1 The camera model in this task compared to full model in lecture have 3 main differences.

- In this calibration assignment we do not consider radial distortion and tangential distortion.
- We do not need to find corner correspondences between image and checkerboard.
- We do not impose constrain to estimate restricted camera model

1.1 Data Normalization

This steps if for avoiding numerical instabilities and it is good to normalize the data.

Q2 Briefly explain the potential problems if we skip this step in our algorithm

A2 There are mainly two reason:

- Data normalization is essential in solving DLT since some entries will take dominant position when solving the minimizing questions. Like the image shown before. The difference in orders of magnitude between column data will lead to poor least-squares result.
- Large entry will be very sensitive to noise.

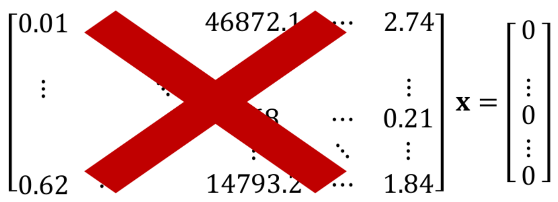


Figure 1: Different magnitude orders between column.

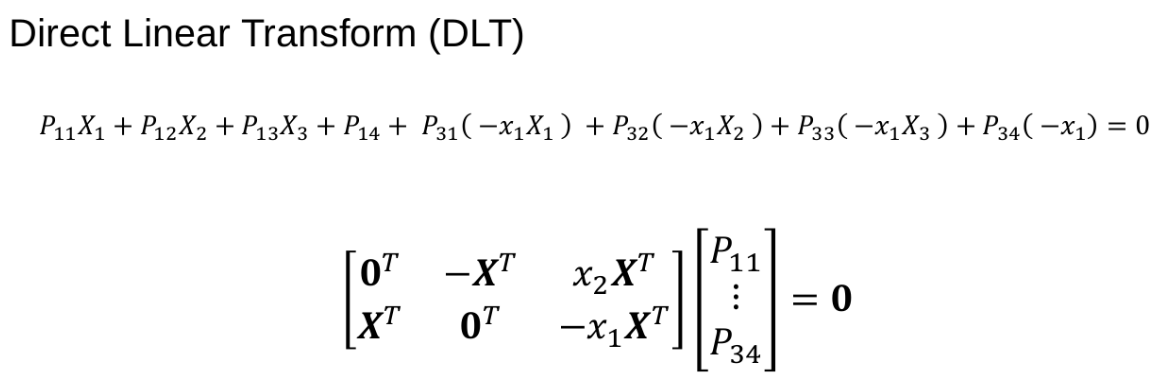
1.2 DLT

DLT algorithm use the basic projection relationships for a 2D point and 3D point to build a constrain matrix

1.2.1 Assembling the constraint matrix

Q3 You need at least 11 independent constraints to determine the 11 degrees of freedom (DoF) of P, but for this exercise just use all available correspondences to minimize the least squares error. How many independent constraints can you derive from a single 2D-3D correspondence?

A3 As shown in figure below. Each 2D-3D correspondences will provide two constraints. But we should attention here the degenerate case like all 3D points are coplane. In this case the 2D-3D correspondence can not provide independent 2 constraints



Direct Linear Transform (DLT)

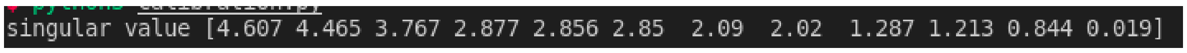
$$P_{11}X_1 + P_{12}X_2 + P_{13}X_3 + P_{14} + P_{31}(-x_1X_1) + P_{32}(-x_1X_2) + P_{33}(-x_1X_3) + P_{34}(-x_1) = 0$$

$$\begin{bmatrix} \mathbf{0}^T & -\boldsymbol{X}^T & x_2\boldsymbol{X}^T \\ \boldsymbol{X}^T & \mathbf{0}^T & -x_1\boldsymbol{X}^T \end{bmatrix} \begin{bmatrix} P_{11} \\ \vdots \\ P_{34} \end{bmatrix} = \mathbf{0}$$

Figure 2: Each 2D-3D correspondences will provide two constraints.

1.2.2 Solving for the nullspace of A

We can find this efficiently using singular value decomposition (SVD). We should attention here. The null space of assembling constraint matrix A is not really the "null space". In this assignment what we get is a over-determinant equation so the smallest singular value is not necessary to be zero like the image shown below. Moreover, the nonzero singular value explains why the algebraic error can be used for optimization.

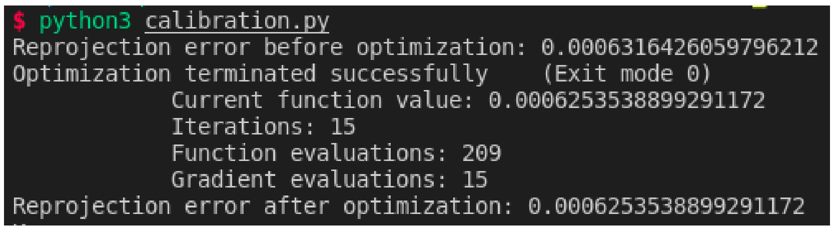


```
singular value [4.607 4.465 3.767 2.877 2.856 2.85  2.09  2.02  1.287 1.213 0.844 0.019]
```

Figure 3: The smallest singular value is 0.019 instead of 0 because of over-determinant.

1.3 Optimizing reprojection errors

Q4 How does the reported reprojection error change during optimization? Discuss the difference between algebraic and geometric error and explain the problem with the error measure e = x × PX in your report.

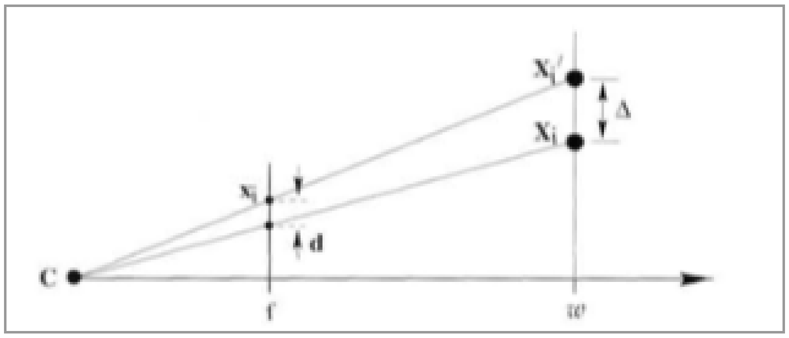


```
$ python3 calibration.py
Reprojection error before optimization: 0.0006316426059796212
Optimization terminated successfully    (Exit mode 0)
            Current function value: 0.0006253538899291172
            Iterations: 15
            Function evaluations: 209
            Gradient evaluations: 15
Reprojection error after optimization: 0.0006253538899291172
```

Figure 4: Optimization with geometric error.

A4 The reprojection error changed from 6.316e-4 to 6.253e-4 after optimization. For the difference between the algebraic error.

- The geometric error is gold-standard since it directly shows the points to points error based on input. It has strong interpretability and moreover, the noise comes mainly from 2D image formation process so it fit well with geometric error. But calculating the reprojection error need to project the 3D points back to 2D so it is more computation expensive.
- The algebraic error focus on minimizing the DLT error $||AP|| = 0$. It is not meaningful and the error will change when the 3D points are further away like the figure shown below. So basically it can not interpretare the error caused by wrong projection only. However, it is computation faster since no extra reprojection is needed.
- The error e = x × PX is problematic since the error will change not only because the x and Px are not co-linear but the magnitude $||x||$ and $||Px||$ will also change the errors.

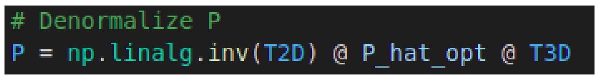


Let X_i be a 3D point and x_i be its observation. The plane w contains X_i and is parallel to the image plane. The algebraic error is Δ, the distance between X_i and the backprojection ray in the plane w. The geometric error d is the distance between x_i and projection of X_i onto the image plane, f. Note that as the 3D point moves farther from the camera, the algebraic error increases, while the geometric error remains constant.

Figure 5: algebraic error changes when geometric error stay the same.

1.4 Denormalizing the projection matrix

This denormalizing process can be easily done with process below:



```
# Denormalize P
P = np.linalg.inv(T2D) @ P_hat_opt @ T3D
```

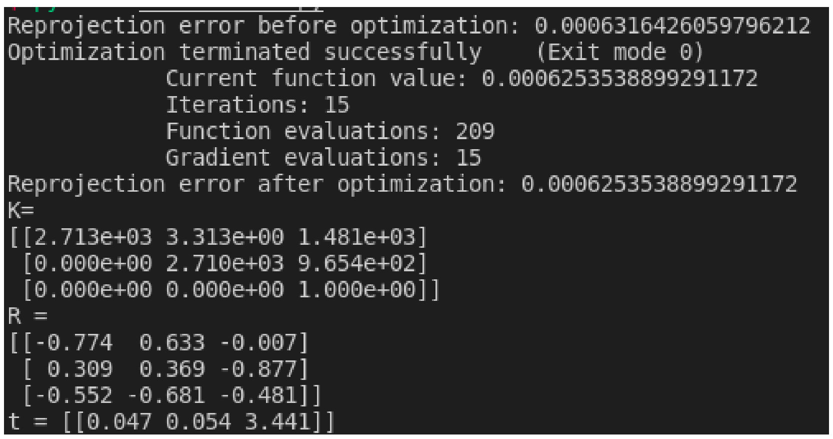
Figure 6: project back to origin image.

1.5 Decomposing the projection matrix

Q5 The position of the camera center C is the nullspace of P and can be used to compute the translation t. Report your computed K, R, and t and discuss the reported re-projection errors before and after nonlinear optimization. Do your estimated values seem reasonable?

A5 The computed K, R and t is shown below:

As we can see the reprojected points is highly close to the 2D image points. The estimated values seem reasonable. The error after optimization decrease 6e-6 which is only 1%. This means the error before the optimization is relatively good.



```
Reprojection error before optimization: 0.0006316426059796212
Optimization terminated successfully    (Exit mode 0)
            Current function value: 0.0006253538899291172
            Iterations: 15
            Function evaluations: 209
            Gradient evaluations: 15
Reprojection error after optimization: 0.0006253538899291172
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]]
```

Figure 7: Computed K,R and t.

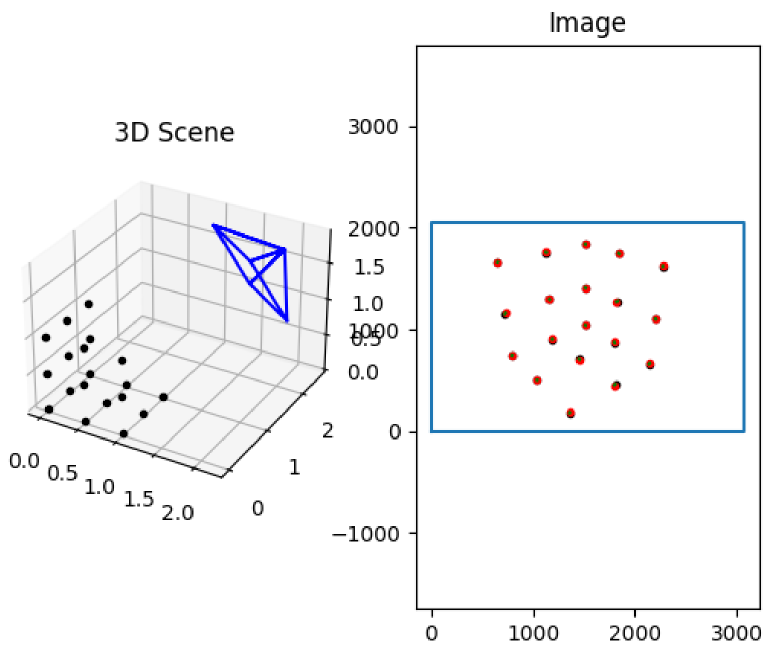


Figure 8: Solution visualization.

2 Structure from Motion

This tasks combine reative pose estimation, absolute pose estimation and point triangulation to a simplified Structure from Motion pipeline.

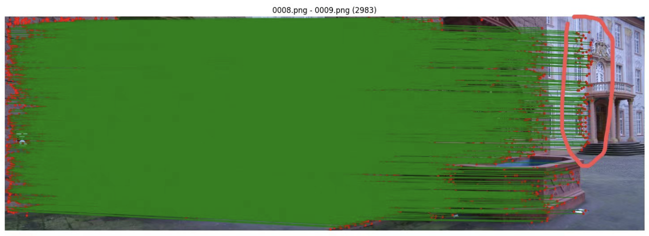
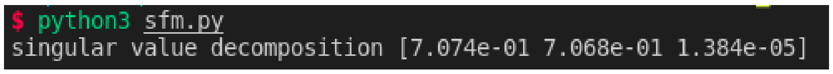


Figure 9: Some far away keypoints correspondence in figure 8-9.

As shown above, there are some far away keypoints in the image which can well expalin the result before.

2.1 Essential matrix estimation

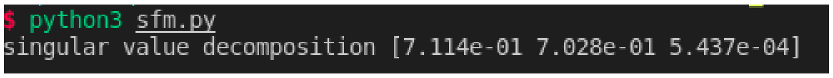
For a proper essential matrix, the first singular values need to be equal (and ¿ 0), and the third one is 0. You can do this by using SVD on your estimate and set the singular values accordingly. Since E is up to scale, you can just set the first singular values to 1. As shown above the computed essential



```
$ python3 sfm.py
singular value decomposition [7.074e-01 7.068e-01 1.384e-05]
```

Figure 10: The first and second singular value is not identical and third is nonzero.

matrix is not so ideal. This is caused by the over-determinant problem and we can force the third entry to be zero. Moreover, we can see this SVD is from init pair [3,4]. What about the error of image pair with larger difference?



```
$ python3 sfm.py
singular value decomposition [7.114e-01 7.028e-01 5.437e-04]
```

Figure 11: The image pair [0,9] have larger error than [3,4].

2.2 Point triangulation

The DLT for point triangulation is already implemented in the framework. However, some points that fulfill the mathematical constraints lie behind one or both cameras. Since these points are clearly wrong, implement a filtering step in the point triangulation.

In this step we ignore the negative-depth points and only keep the positive depth value points

2.3 Finding the correct decomposition

To find the actually correct one, try to triangulate points with each one and use the one with the most points in front of both cameras.

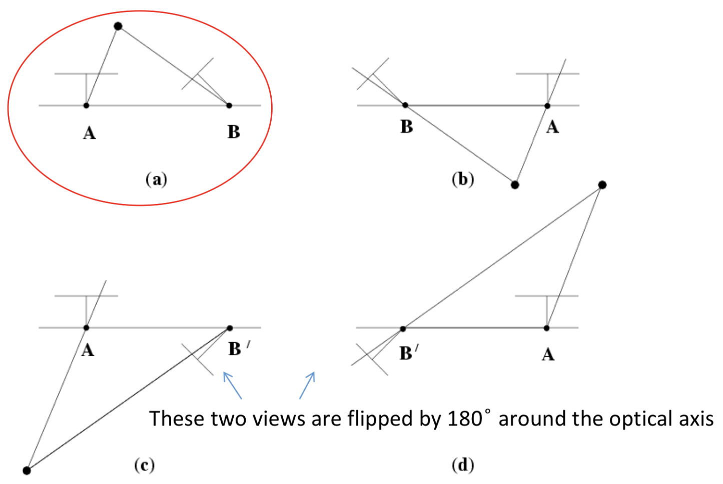


Figure 12: In total 4 candidate solution so we need to find the correct one.

Even though some points may be triangulated behind the camera image plane but the correct decomposition should have most points in front of both camera.

2.4 Absolute pose estimation

Given a point cloud and correspondences to the image keypoints we can estimate the absolute pose of a new image with respect to the scene.

2.5 Map extension

Once we know the new images pose we can extend the map by triangulating new points from all possible pairs of registered images. Implement the triangulation with all pairs and make sure to correctly add the new 2D-3D correspondences to each image.

2.6 Analysis

In this section we will discuss the result of SFM: First we will show the best result from our algorithm which use pair [5,6] and the column far away is easy to see

After this we will focus on the error due to bad initialization. First the figure belows is the result be init pair [3,4].

As it discussed in Moodle. The init [3,4] will bring a bad result as figures shows:

In order to analyze the bad result we shows the reprojection error for each poses. As we can see the error become bigger and bigger as the sequence goes. It's easy to understand beccause in SFM the error will always propagate to next poses. This error is actually introduced by new 3D points which is triangulated with new poses and former poses. Because the poses is shift away so the error will also shift away. In order to prove this we can only use the keypoint and 3D points from the origin pair [3,4] and do not add new points.

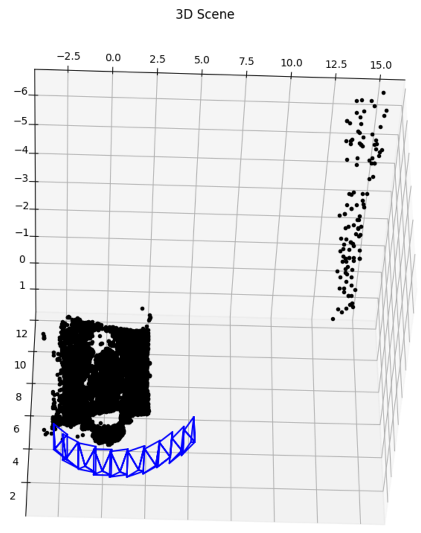


Figure 13: One column is far away which is bad for visualization.

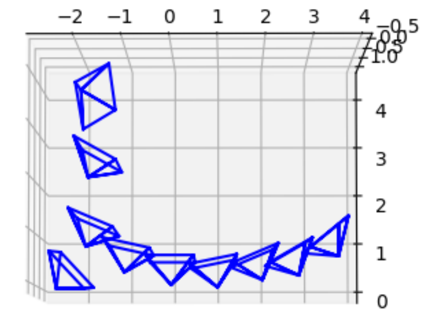
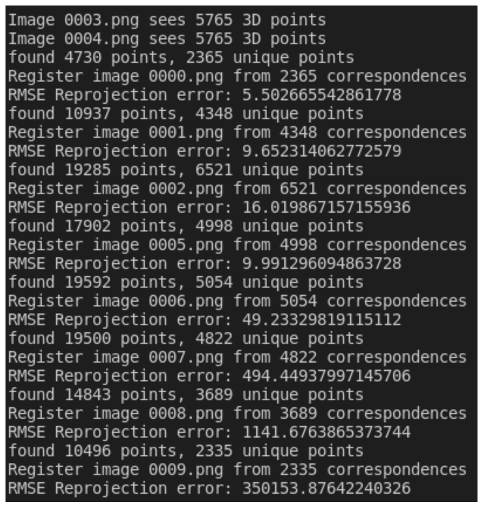


Figure 14: Camera pose result seems bad.



```
Image 0003.png sees 5765 3D points
Image 0004.png sees 5765 3D points
found 4730 points, 2365 unique points
Register image 0000.png from 2365 correspondences
RMSE Reprojection error: 5.502665542861778
found 10937 points, 4348 unique points
Register image 0001.png from 4348 correspondences
RMSE Reprojection error: 9.652314062772579
found 19285 points, 6521 unique points
Register image 0002.png from 6521 correspondences
RMSE Reprojection error: 16.019867157155936
found 17902 points, 4998 unique points
Register image 0005.png from 4998 correspondences
RMSE Reprojection error: 9.991296094863728
found 19592 points, 5054 unique points
Register image 0006.png from 5054 correspondences
RMSE Reprojection error: 49.23329819115112
found 19500 points, 4822 unique points
Register image 0007.png from 4822 correspondences
RMSE Reprojection error: 494.44937997145706
found 14843 points, 3689 unique points
Register image 0008.png from 3689 correspondences
RMSE Reprojection error: 1141.6763865373744
found 10496 points, 2335 unique points
Register image 0009.png from 2335 correspondences
RMSE Reprojection error: 350153.87642240326
```

Figure 15: The sequential SFM will have larger and larger reprojection error

Since the problem comes from the inaccurate 3D points in the SFM sequences this problem can be resolved by some methods.

- First we can change the init which have more accurate 3D points. Like pair [5,6]
- We can also try large origin 3D points pair since large baseline will have accurate 3D position estimation. Like the figure shows below:
- We can correct the shift by global SFM instead of sequential SFM

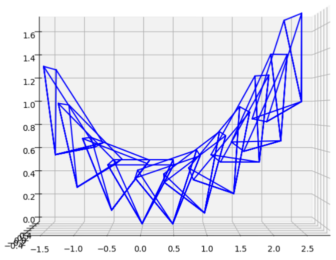


Figure 16: Only estimate pose with origin point from pair [3,4] actually works

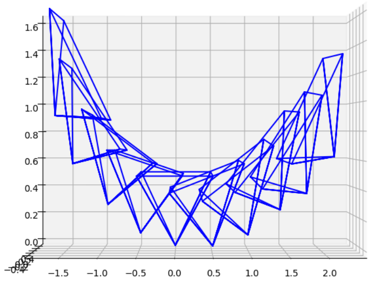


Figure 17: Large baseline [3,5] works