

Computer Vision Lecture

Assignment 3 Report

Autumn Term 2022

Contents

1	Report	2
1.1	Camera Calibration	2
1.2	SFM	5

Chapter 1

Report

1.1 Camera Calibration

Brief Explanation of the approach

Given are 3D-2D correspondences. This means we know for certain image points the corresponding 3D points. Given the projection equation (see fig. 1.1) one can find the P matrix with the DLT. What we basically do is rewriting the projection equation to: see 1.2. We get 2 equations per correspondence. In total we have 11 unknowns. Thus we need 6 point correspondence at least. Since we have more here we find the least square solution through singular value decomposition. What we then find is $P = K[R|t]$. Next we need to decompose P to get R, K using QR decomposition of P and ensuring K has pos. diagonal elements and R has $\det(R)=1$. t is obtained as the nullspace of P. Note that I use normalized image coordinates in the range of $[-1,1]$. Now we have everything we need K, R and t.

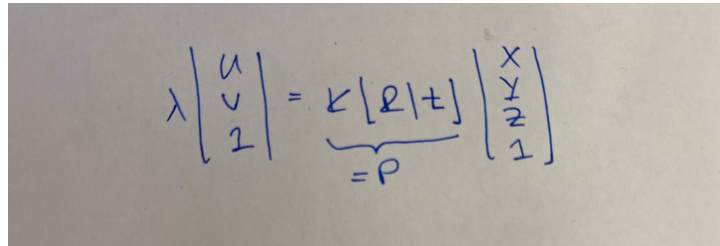

$$\lambda \begin{pmatrix} u \\ v \\ 1 \end{pmatrix} = \underbrace{K[R|t]}_{=P} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix}$$

Figure 1.1: Projection Equation

Which part of the full model is not calibrated with our approach

Assuming the full model also includes the distortion. We do not calibrate the distortion since we don't take it into account. It could be taken into account when we optimize the re-projection error. The solution from the linear DLT approach could be used as a initial guess - that's what we already do but we don't optimize for the camera distortion parameters.

Potential problem without normalization

Without normalization there are orders of magnitude difference between columns in the P matrix. This means our P matrix is ill-conditioned being very sensitive to noise. This means a small change in the input causes a huge effect on the output.

$$\underbrace{\begin{pmatrix} X_w^1 & Y_w^1 & Z_w^1 & 1 & 0 & 0 & 0 & -u_1 X_w^1 & -u_1 Y_w^1 & -u_1 Z_w^1 & -u_1 \\ 0 & 0 & 0 & 0 & X_w^1 & Y_w^1 & Z_w^1 & -v_1 X_w^1 & -v_1 Y_w^1 & -v_1 Z_w^1 & -v_1 \\ & & & & & & & & & & \vdots \\ X_w^n & Y_w^n & Z_w^n & 1 & 0 & 0 & 0 & -u_n X_w^n & -u_n Y_w^n & -u_n Z_w^n & -u_n \\ 0 & 0 & 0 & 0 & X_w^n & Y_w^n & Z_w^n & -v_n X_w^n & -v_n Y_w^n & -v_n Z_w^n & -v_n \end{pmatrix}}_{\text{Q (this matrix is known)}} \underbrace{\begin{pmatrix} m_{11} \\ m_{12} \\ m_{13} \\ m_{14} \\ m_{21} \\ m_{22} \\ m_{23} \\ m_{24} \\ m_{31} \\ m_{32} \\ m_{33} \\ m_{34} \end{pmatrix}}_{\text{M (this matrix is unknown)}} = \begin{pmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ 0 \end{pmatrix} \Rightarrow \mathbf{Q} \cdot \mathbf{M} = \mathbf{0}$$

Figure 1.2: DLT

As a result, least squares yields poor results. Thus, it's important that we normalize it by transforming image coordinates to the range of $[-1,1]$.

Number of ind. constraints for single 2D-3D correspondence

2, the Q matrix has 3 rows for one correspondence but is only rank 2. Thus, only two equations and not three.

Optimizing re-projection errors

How does the reported re-projection error change during optimization?

The re-projection error gradually decreases during optimization i.e. until the difference of two consecutive iterations is lower than the tolerance specified for the optimizer.

Difference between algebraic and geometric error?

The algebraic error is $e_{alg} = x \times PX$. The algebraic error is zero when the two lines are parallel. One drawback of this error is that it increases for points far away. In contrast, the geometric error (also known as re-projection error) does not. Thus, if we have points close to the camera and points far away in the image this could be a problem since distant points would be overemphasized. Furthermore, the algebraic error can be minimized with SVD in polynomial time whereas the geometric error uses e.g. gradient based methods and thus its computation time increases more with more points. If we had many points and we care about speed i.e. online calibration one might consider the algebraic error. The geometric error minimizes the squared distance of the pixels and the re-projected pixels.

Problem with error measure $e = x \times PX$

As outlined above: One problem is that distant points are overemphasized in the algebraic error. Whereas the geometric error only takes the direction into account.

Results

See fig. 1.3.

```

Reprojection error after optimization: 0.0006253538899281121
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 1.3: Results of camera calibration

Are the estimated values reasonable?

Yes, looking e.g. at t_z we see the camera is approx. 3.44 meters away that seems to make sense. The re-projection error is also ≈ 1 . Looking at the generated figure below (1.4) we see the camera looking at the key points as given images imply.

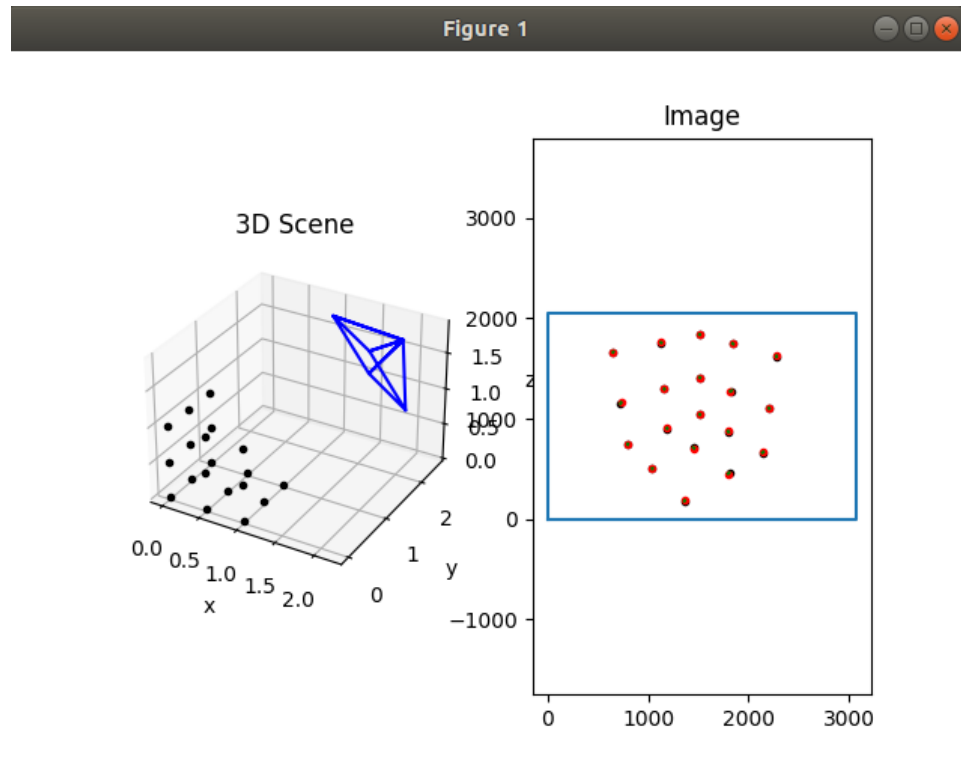


Figure 1.4: Results visualized

1.2 SFM

First, we find the relative pose of two images through the epipolar constraint. To get there one takes the epipolar constraint i.e. $x_1^T E x_2 = 0$ and then applies DLT which means re-writing the equation to: see 1.5. Note that x_1 and x_2 are two corresponding image points. This means for the epipolar constraint we need a 2D-2D correspondence. Note: In the epipolar constraint we use normalized image coordinates i.e. $x_1 = K^{-1} * x_1$. This way

$$\underbrace{\begin{bmatrix} \bar{u}_2^1 \bar{u}_1^1 & \bar{u}_2^1 \bar{v}_1^1 & \bar{u}_2^1 & \bar{v}_2^1 \bar{u}_1^1 & \bar{v}_2^1 \bar{v}_1^1 & \bar{v}_2^1 & \bar{u}_1^1 & \bar{v}_1^1 & 1 \\ \bar{u}_2^2 \bar{u}_1^2 & \bar{u}_2^2 \bar{v}_1^2 & \bar{u}_2^2 & \bar{v}_2^2 \bar{u}_1^2 & \bar{v}_2^2 \bar{v}_1^2 & \bar{v}_2^2 & \bar{u}_1^2 & \bar{v}_1^2 & 1 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \bar{u}_2^n \bar{u}_1^n & \bar{u}_2^n \bar{v}_1^n & \bar{u}_2^n & \bar{v}_2^n \bar{u}_1^n & \bar{v}_2^n \bar{v}_1^n & \bar{v}_2^n & \bar{u}_1^n & \bar{v}_1^n & 1 \end{bmatrix}}_{\text{Q (this matrix is known)}} \underbrace{\begin{bmatrix} e_{11} \\ e_{12} \\ e_{13} \\ e_{21} \\ e_{22} \\ e_{23} \\ e_{31} \\ e_{32} \\ e_{33} \end{bmatrix}}_{\bar{E} \text{ (this matrix is unknown)}} = 0$$

Figure 1.5: Rewritten epipolar constraint

The E-matrix is the nullspace of the Q matrix which can be found through SVD. The E matrix is a product of a rotation (R) and a skew-symmetric matrix. This fact implies the internal constraints of the essential matrix. The internal constraint means that the first two singular values of E must be equal and the third one zero. This is because R doesn't change the singular values and the singular values of the skew-symmetric matrix $[t]_{\times}$ are s,s,0. Thus the singular values of E must also be s,s,0. Since E is only defined up to scale we choose s to be 1. Afterwards the E matrix needs to be decomposed into R and t using SVD. R and t characterizes the relative position of the two initial images. Important here: This decomposition will result in 4 different solutions. The best solution is the one with all/most points in front of the camera i.e. $z > 0$. Once we have R and t we can finally triangulate the first 3D points. Recall: So far we made use of 2D-2D correspondences and the DLT.

Now, we also make use of the obtained 2D-3D correspondence. We take the next picture. Then, we first find the matching keypoints (already given to us) of the newly added image with all previously added images. Next, we check which matching keypoints are already associated with a 3D point. Based on this obtained 2D-3D correspondence we can calculate the relative pose of the new image w.r.t. to the initial image using projection equation with DLT.

The goal of the next step is to also triangulate the matching keypoints which do not have a 3D point associated yet. If we did not do that we would not have enough matching keypoints with associated 3D points when we add the next image since we move around the object. Thus, we would not be able to reliably estimate the relative pose of the next images.

The matching keypoints without associated 3D points are triangulated again using the R and t which was computed previously. Now, all matched keypoints have an associated 3D point. This procedure is repeated until we looped over all images.