Computer Vision Lab 3 Report

Jingyan Li

1. CALIBRATION

In the calibration, I firstly conducted data normalization for input 2D points and 3D points. Then I calculated the projection matrix P by DLT. Since it is only the rough estimation of P, I later optimized reprojection error (i.e. algebraic error) to optimize P. Finally we denormalized P such that it can be applied to coordinates without normalization. In the end, we also decomposed P to get the camera metrics: intrinsic matrix, camera center and translation matrix.

1.1 Which part of the full model is not calibrated?

In our approach, we cannot calibrate the coplanar objects. For example the bottom plan in X-Y are shown as colinear in the 2D image, which cannot be calibrated.

1.2 Data Normalization

If we skip the normalization, the points vary a lot and we face some extreme values in the coordinate list. It makes inconsistent scales between points. After doing the normalization, we need to return back to the original projection matrix, such that we can directly transfer the original 3D coordinate to 2D coordinate before normalization. That is the reason why we need to keep the transformation matrix.

1.3 DLT

There should be 6 independent constraints that we derive from a single 2D-3D correspondence.

Suppose that $k \in \{1, ..., N\}$. Let $\mathbf{x}_k = (x_{1k}, x_{2k}) \in \mathbb{R}^2$ and $\mathbf{y}_k = (y_{1k}, y_{2k}, y_{3k}) \in \mathbb{R}^3$, and in DLT we want to find 2×3 matrix A such that

$\alpha_k \, \mathbf{x}_k = \mathbf{A} \, \mathbf{y}_k$

We can use $\mathbf{H} = \begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix}$ to help us solve the equation.

$(\mathbf{x}_k^T \, \mathbf{H}) \, \alpha_k \, \mathbf{x}_k = (\mathbf{x}_k^T \, \mathbf{H}) \, \mathbf{A} \, \mathbf{y}_k$

$\alpha_k \, \mathbf{x}_k^T \, \mathbf{H} \, \mathbf{x}_k = \mathbf{x}_k^T \, \mathbf{H} \, \mathbf{A} \, \mathbf{y}_k$

$\mathbf{x}_k^T \, \mathbf{H} \, \mathbf{A} \, \mathbf{y}_k = 0$

Let $\mathbf{x_k} = \begin{pmatrix} x_{1k} \\ x_{2k} \end{pmatrix}$, $\mathbf{y_k} = \begin{pmatrix} y_{1k} \\ y_{2k} \\ y_{3k} \end{pmatrix}$, and $x_k A y_k = 0$. Then $A = \begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \end{pmatrix}$. Then we have one equation of six unknowns.

1.4 Reprojection Errors

Errors firstly went down and then converged at 0.000625. The two errors are a geometric error function which measures the distance between the 3D point's projection and the 2D observation, and an algebraic error function, which is the residual of a homogeneous least-squares problem respectively. If we assume noise are the most sensible in the majority of cases, geometric solution should be the "right" solution. This is exactly what we want to achieve in the task. In contrast, roughly speaking, the algebraic error measures the distance between the known 3D point and the observation's backprojection ray. This implies errors arise from noise in 3D points as opposed to the camera itself, and tends to overemphasize distant points when finding a solution. However, the "right" solution is a cost of time. The geometric solution's cost function grows linearly with the number of observations, whereas the algebraic cost function is constant (after an SVD operation in preprocessing).

jingyli@ethz.ch

1.5 Decompose the projection matrix

The reprojection errors before optimization is 0.000632, and went down to 0.000625 after optimization. The final result is shown below.

$$R = \begin{pmatrix} -0.774 & 0.633 & -0.007 \\ 0.309 & 0.369 & -0.877 \\ -0.552 & -0.681 & -0.481 \end{pmatrix},$$

$$K = \begin{pmatrix} 2713 & 3.313 & 1481 \\ 0 & 2710 & 965.4 \\ 0 & 0 & 1 \end{pmatrix},$$

$$t = \begin{pmatrix} 0.047 & 0.054 & 3.441 \end{pmatrix}$$

The estimated result is shown in Figure 1. As we can see, the original points and reprojected points align well in the estimation.

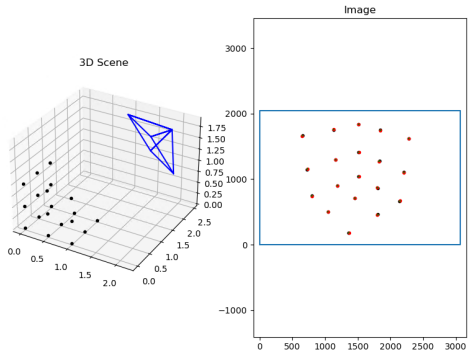

Figure 1. The estimated result of calibration.

2. STRUCTURE FROM MOTION

In this section, we first estimated essential matrix by a similar procedure as we did in calibration (i.e., using DLT). Then we conducted point triangulation to use 2D image points as well as pose information to reconstruct 3D points. Applying the inverse of the camera matrix on an image point gave us a ray passing through the point. After acquiring the ray of the corresponding point in the second view, we can find the the 3D point by finding the point of intersection of the two rays. Then we selected the correct pose which contains the most point in front of the camera. Then we estimated the pose of new images and extended the map to all registered images.