EEN020 Computer Vision

Yuhui Bi

19991021-0518

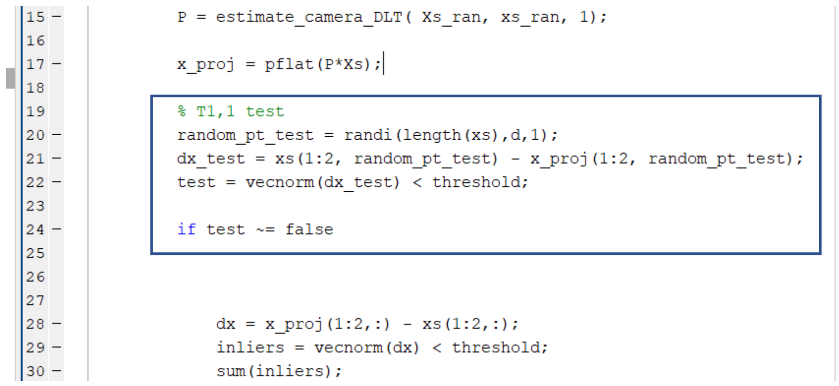
2 January 2023

Algorithm design:

Starting with *get_dataset_info* and *imread* function to access different images, we can select first 2 initial pairs manually to construct matches and *desc_X,* we can obtain initial 2D points xA and xB of 2 images. Then we can estimate essential matrix E robustly, when we have enough inliers. Previous 2D points need to be normalized by multiplying inverse of K, and we need to filter x_norm and desc_X by sequence of inliers. Next we triangulate 3D points by DLT method, when we have 3D points X, we see whether all points are in front of both cameras, if they are in front of both cameras then we use this set out of 4 sets (discarding rest 3). This is important to pick initial 2 cameras. Note that we need to filter X again by discarding long distance points as required. Then by *vl_sift*, we can obtain matches of different 2D points with common 3D points. From the matched ones, we can obtain different 2D points of the rest of the images. Normalising again, we can robustly estimate different cameras P. We need to skip 2 cameras of initial pairs, since their P are calculated before already. Then we can first plot all images and cameras. One more step left to do is bundle adjustment. We need to extend matched 2D points to full size to match with 3D points, so those invisible points are set to 0. Because the full data set is huge and takes forever to run, sometimes we just split partial data (3D points and normalized 2D points) to run bundle adjustments over both 3D points and cameras. Then we can visulise them in terms of before and after BA.

One point to mention is that the code takes a super long time to run, so I just save some data occasionally, start with a new script and develop and verify there; When it is good enough, I bring it back to *run_Sfm.m*.

The most challenging part is to develop robust estimation on P and bundle adjustment.
For robust P, we can start with set epsilon and alpha, and calculate T with equations as before. Then we do it inside iterations, when it meets the requirements, break it, and record it. Starting with some random points, we first estimate camera P by DLT, project 3D points into calculated P, and see whether the distance between projected 2D points and real points is small enough (smaller than the threshold). If so, we update parameters and record the sequence of inliers and use current P as the best P, then we iterate again until the loop breaks.
One optional question is solved here as well: $T_{1,1}$ test. The essential is to modify the current RANSAC for some place and check if one data point is inlier quickly. If not, discard it. So, we can select one random point from the 2D point and projected 2D point respectively, and see whether the norm of a vector (from the projected one to the real one) is smaller than the threshold. If so we can continue, otherwise, we discard it and do another iteration. This is called: early bailout: $T_{d,d}$ test, as seen below, labelled in the box, (where d=1, set before).



```
P = estimate_camera_DLT( Xs_ran, xs_ran, 1);

x_proj = pflat(P*Xs);

% T1,1 test
random_pt_test = randi(length(xs),d,1);
dx_test = xs(1:2, random_pt_test) - x_proj(1:2, random_pt_test);
test = vecnorm(dx_test) < threshold;

if test ~= false



    dx = x_proj(1:2,:) - xs(1:2,:);
    inliers = vecnorm(dx) < threshold;
    sum(inliers);
```

For bundle adjustment, my codes from HA5 are obviously not enough since we need to optimize both 3D points (I have) and cameras (I don't have). Starting from a low number of iterations and high lambda, we can compute the initial error, then we do it inside iterations. First, we linearize reproject error with 3D points to calculate residual (refer to equation 12) and Jacobian matrix (refer to equation 14) (same as before), then we calculate the delta and update 3D points. Then we need to linearize reprojection error for cameras, and then update cameras. This part of residual calculations is the same but another different Jacobian (refer to equation 22). Lastly, we keep recording errors. We do it with set iterations.

$$\sum_{i=1}^{m}\sum_{j=1}^{n}\|r_i(\mathbf{X}_j)\|^2 = \sum_{i=1}^{m}\sum_{j=1}^{n}\left\|\left(x_{ij}^1 - \frac{P_i^1\mathbf{X}_j}{P_i^3\mathbf{X}_j},\ x_{ij}^2 - \frac{P_i^2\mathbf{X}_j}{P_i^3\mathbf{X}_j}\right)\right\|^2, \tag{12}$$

$$J_i(\mathbf{X}_j) = \begin{bmatrix} \frac{(P_i^1\mathbf{X}_j)}{(P_i^3\mathbf{X}_j)^2}P_i^3 - \frac{1}{P_i^3\mathbf{X}_j}P_i^1 \\ \\ \frac{(P_i^2\mathbf{X}_j)}{(P_i^3\mathbf{X}_j)^2}P_i^3 - \frac{1}{P_i^3\mathbf{X}_j}P_i^2 \end{bmatrix}. \tag{14}$$

$$J_{ij}^{\mathbf{P}_i}(P_i,\mathbf{X}_j) = \begin{bmatrix} -\frac{1}{P_i^3\mathbf{X}_j} & 0 & \frac{(P_i^1\mathbf{X}_j)}{(P_i^3\mathbf{X}_j)^2} \\ 0 & -\frac{1}{P_i^3\mathbf{X}_j} & \frac{(P_i^2\mathbf{X}_j)}{(P_i^3\mathbf{X}_j)^2} \end{bmatrix}(\mathbf{X}_j^T \otimes I_3). \tag{22}$$

where $(\mathbf{X}_j^T \otimes I_3)$ is a 3×12 matrix corresponding tos the Kronecker product between $\mathbf{X}_j^T$ and the 3×3 identity matrix I_3 (see Matrix Cookbook if you are not familiar with the Kronecker product).

Datasets and plots:

Dataset 1:
This is the easiest dataset, with only 2 images and 2 cameras. As seen in fig 1.1-1.3. All points and cameras are plotted. Bundle adjustment is not that obvious, basically invisible. It would be better if one more camera and image are provided (I download more images and extend this later for optional points).

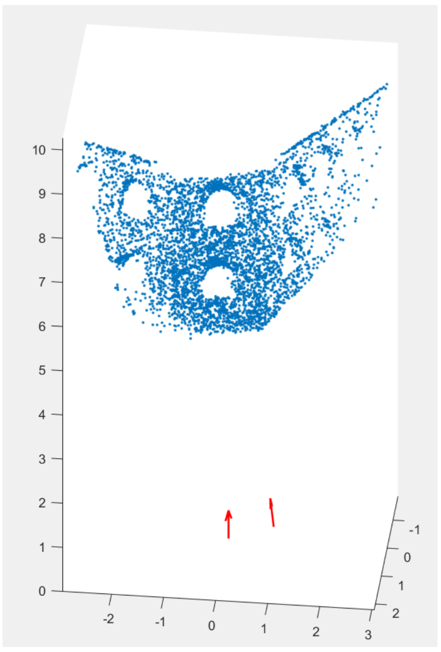


Figure 1.1: All points and all cameras after robustly estimation of cameras (Dataset 1).

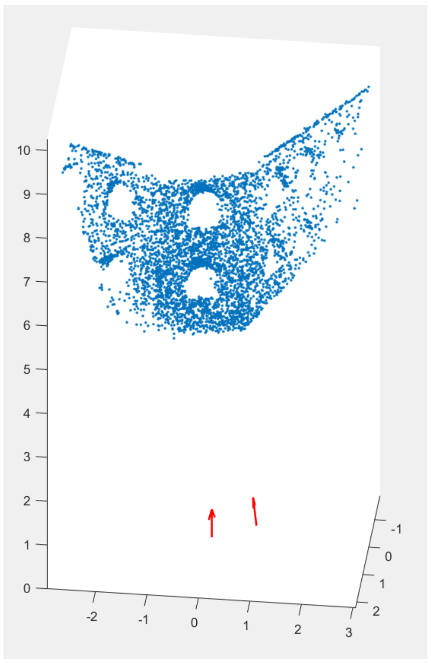


Figure 1.2: All points and all cameras before bundle adjustment (Dataset 1).

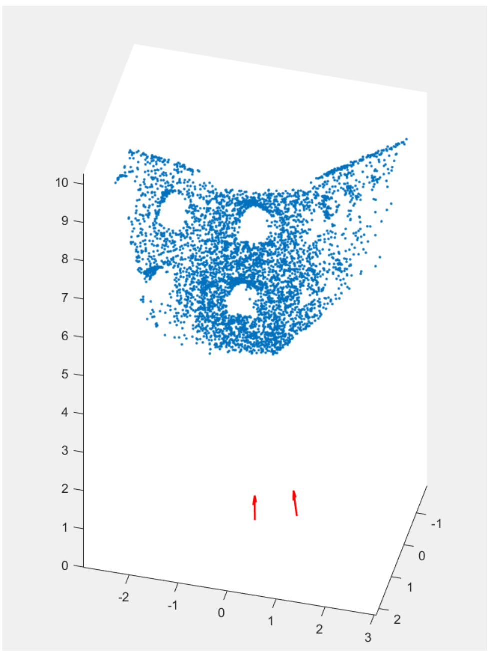


Figure 1.3: All points and all cameras after bundle adjustment (Dataset 1).

Dataset 2:

There are 9 images and 9 cameras. As shown in fig 2.1-2.3, all points and all cameras are shown in fig 2.1. Then I select partial points for faster bundle adjustment comparison. Before and after BA can be seen in fig 2.2-2.3. Points and cameras are moved for short distances.

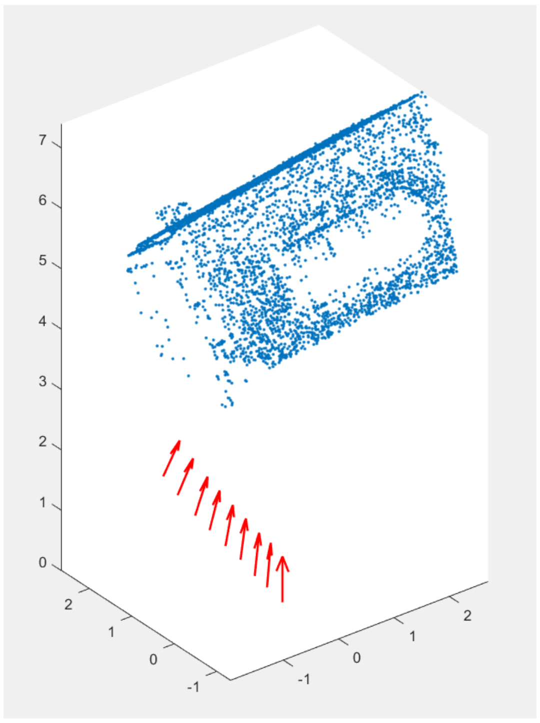


Figure 2.1: All points and all cameras after robustly estimation of cameras (Dataset 2).

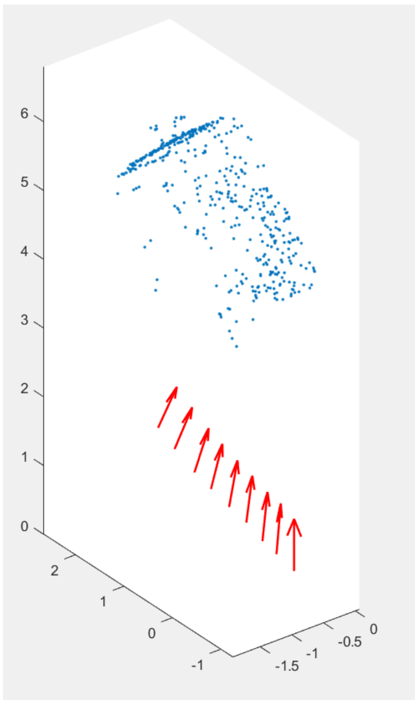


Figure 2.2: Partial points and all cameras before bundle adjustment (Dataset 2).

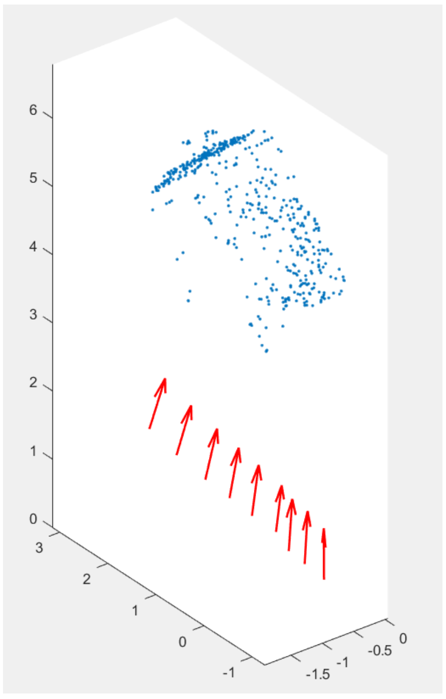


Figure 2.3: Partial points and all cameras after bundle adjustment (Dataset 2).

Dataset 3:
There are 12 images and 12 cameras. As shown in fig 3.1-3.3, all points and all cameras are shown in fig 3.1. Then I select partial points for faster bundle adjustment comparison. Before and after BA can be seen in fig 3.2-3.3. Points and cameras are moved for short distances.

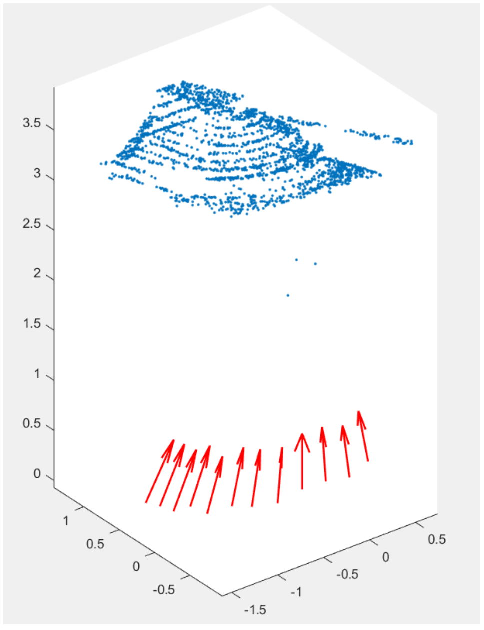


Figure 3.1: All points and all cameras after robustly estimation of cameras (Dataset 3).

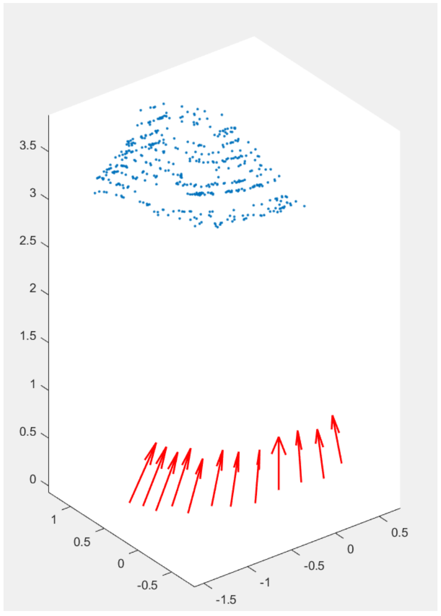


Figure 3.2: Partial points and all cameras before bundle adjustment (Dataset 3).

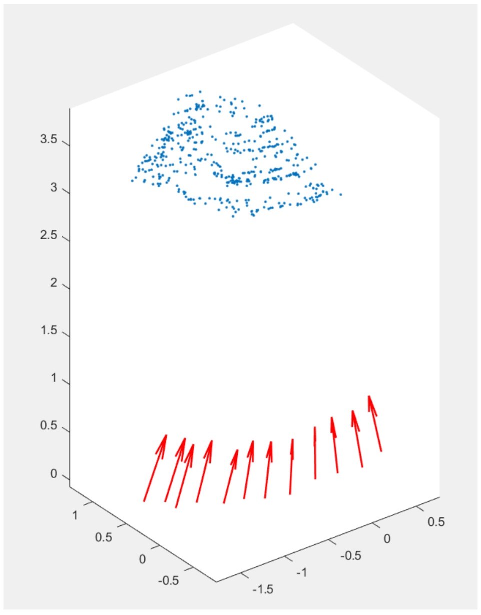


Figure 3.3: Partial points and all cameras after bundle adjustment (Dataset 3).

Dataset 4:

There are 14 images and 14 cameras. As shown in fig 4.1-4.3, all points and all cameras are shown in fig 4.1. Then I select partial points for faster bundle adjustment comparison. Before and after BA can be seen in fig 4.2-4.3. Points and cameras are moved for short distances.

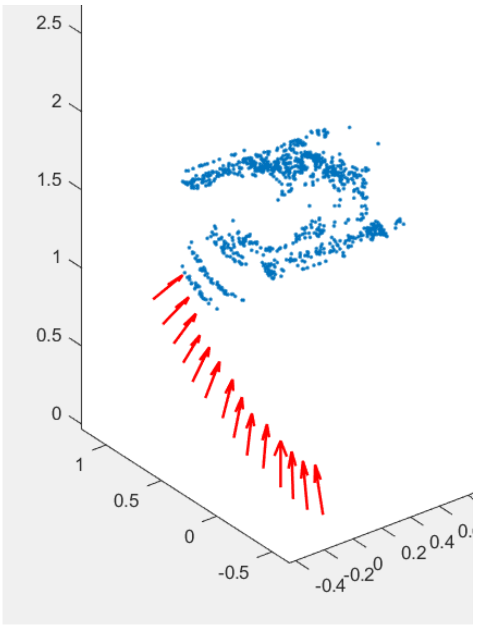


Figure 4.1: All points and all cameras after robustly estimation of cameras (Dataset 4).

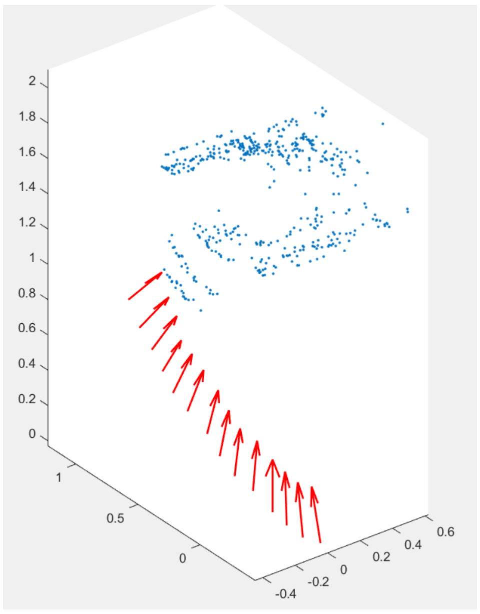


Figure 4.2: Partial points and all cameras before bundle adjustment (Dataset 4).

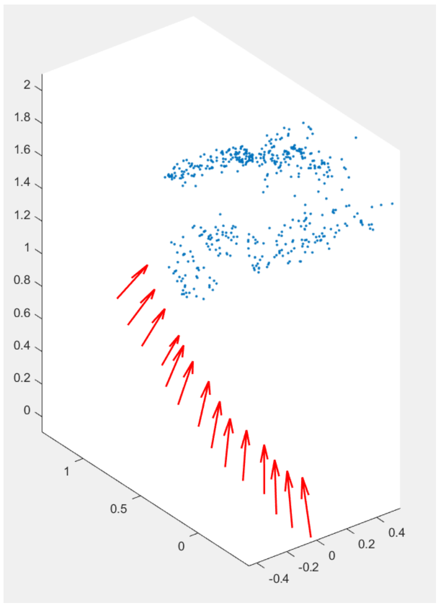


Figure 4.3: Partial points and all cameras after bundle adjustment (Dataset 4).

Dataset 5:

This is the most difficult dataset. It takes me some time to run until the following realization. If more time is allowed, I will consider making it better, since some cameras are placed in weird places. For this dataset, my software's performance is not as good as before. As seen in fig 5.1-5.3. All points and cameras are plotted in fig 5.1. Bundle adjustment is basically moving some 3D points and some cameras for short distances, as seen in fig 5.2-5.3.

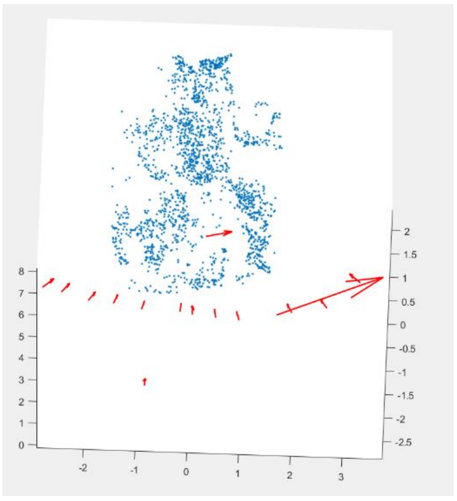


Figure 5.1: All points and all cameras after robustly estimation of cameras (Dataset 5).

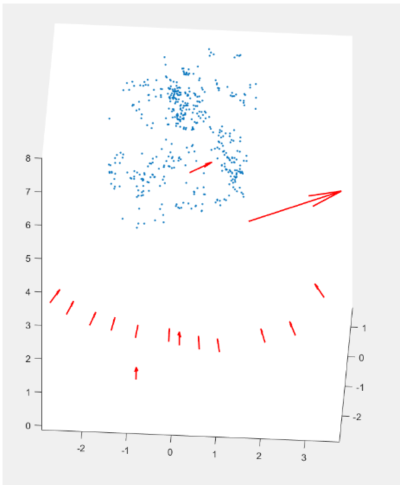


Figure 5.2: Partial points and all cameras before bundle adjustment (Dataset 5).

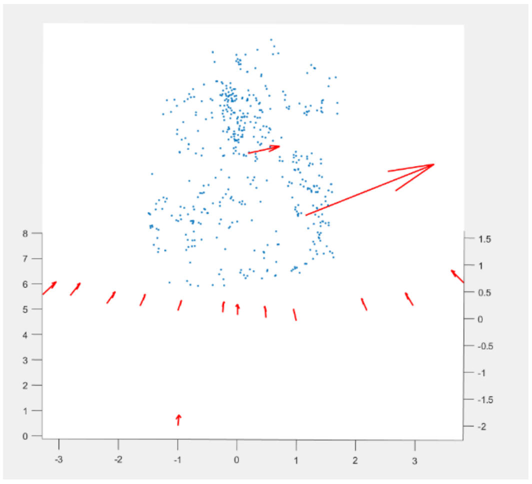


Figure 5.3: Partial points and all cameras after bundle adjustment (Dataset 5).

Choice of initial pairs:

As seen in fig 6.1, assuming we have an orange object and cameras P1-P7, one student from the course of SSY098 Image analysis, suggested I choose cameras from the middle left and the middle right respectively, which are P3 and P5, as shown in fig 6.2. To choose initial pairs, don't choose central cameras, don't choose the furthest cameras, we can pick two, one from the middle left, and one from the middle right. They shouldn't be too far. Please note that minor changes may happen when selecting manually.

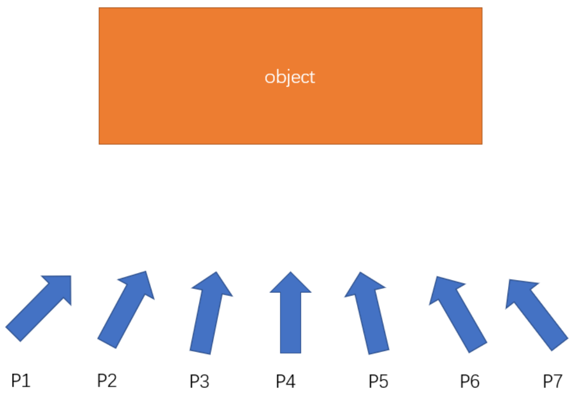


Fig 6.1: Illustration of cameras and the object

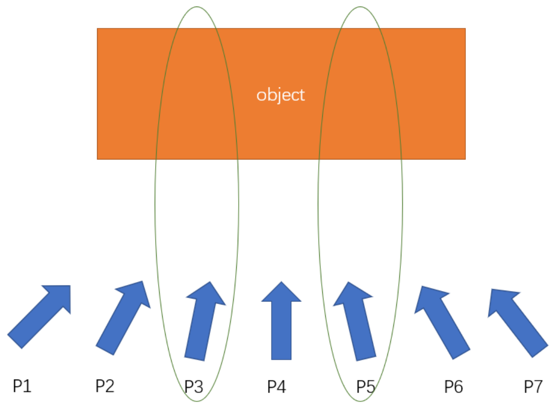
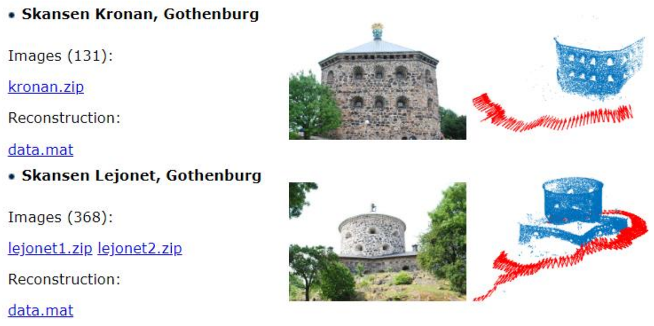


Fig 6.2: Illustration of selection on initial pair of cameras

In this way, we can modify *get_dataset_info* to initial pairs manually.

Other datasets:
Christopher suggested I to find more datasets from LTH:
https://www.maths.lth.se/matematiklth/personal/calle/dataset/dataset.html
I have tried many statues and buildings, but only the following two can achieve a good performance. Dataset 1 is Skansen Kronan, but with only 2 images, here I will use multiple images for dataset 6.1 and 6.2. Dataset 7 is for Skansen Lejonet.



- **Skansen Kronan, Gothenburg**

Images (131):

kronan.zip

Reconstruction:

data.mat

- **Skansen Lejonet, Gothenburg**

Images (368):

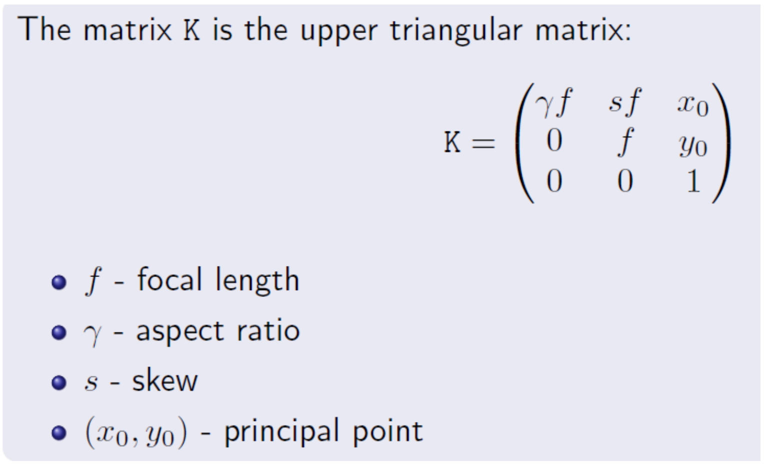
lejonet1.zip lejonet2.zip

Reconstruction:

data.mat

Inner parameter matrix:

From lecture 3, we learn below:


The matrix K is the upper triangular matrix:

$$K = \begin{pmatrix} \gamma f & sf & x_0 \\ 0 & f & y_0 \\ 0 & 0 & 1 \end{pmatrix}$$

- f - focal length
- γ - aspect ratio
- s - skew
- (x_0, y_0) - principal point

We always set skew to 0 and aspect ratio to 1. From focal length and image width and image height, we can compute K matrix.
In code, it is:

```
K = [focal_length 0 im_width/2; 0 focal_length im_height/2; 0 0 1]
```

Dataset 6.1 and 6.2 (Skansen Kronan):
Dataset 6.1 is from Skansen Kronan. For dataset 6.2, I choose another 10 images from Skansen Kronan which are different from dataset 6.1.
There are 10 images and 10 cameras. As shown in fig 6.1.1-6.1.3, all points and all cameras are shown in fig 6.1.1. Then I select partial points for faster bundle adjustment comparison. Before and after BA can be seen in fig 6.1.2-6.1.3. Points and cameras are moved for short distances.

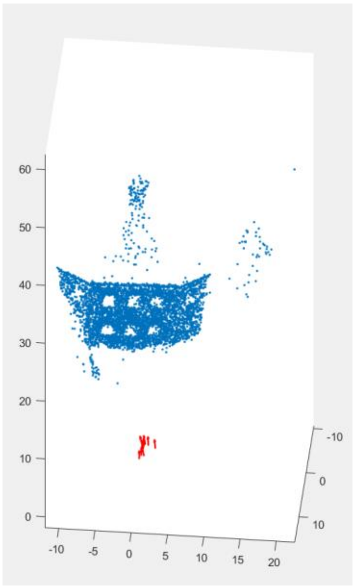


Figure 6.1.1: All points and all cameras after robustly estimation of cameras (Dataset 6.1).

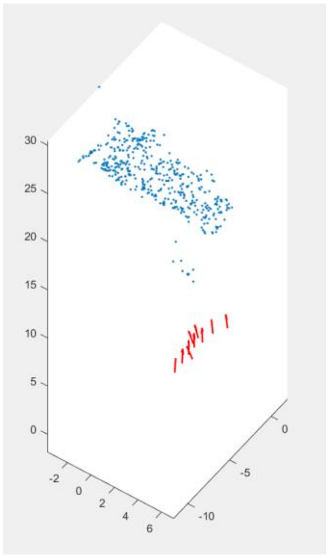


Figure 6.1.2: Partial points and all cameras before bundle adjustment (Dataset 6.1).

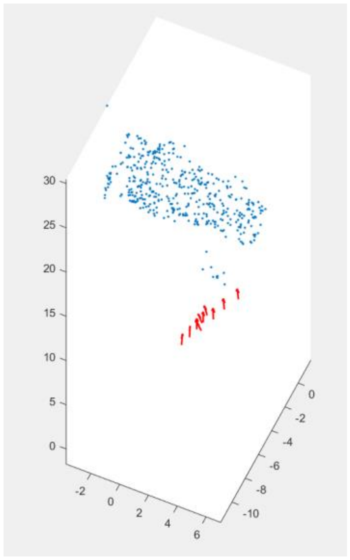


Figure 6.1.3: Partial points and all cameras after bundle adjustment (Dataset 6.1).

Afterward, there are 10 images and 10 cameras. As shown in fig 6.2.1-6.2.3, all points and all cameras are shown in fig 6.2.1. Then I select partial points for faster bundle adjustment comparison. Before and after BA can be seen in fig 6.2.2-6.2.3. Points and cameras are moved for short distances. As seen, performance here is worse compared with dataset 6.1.

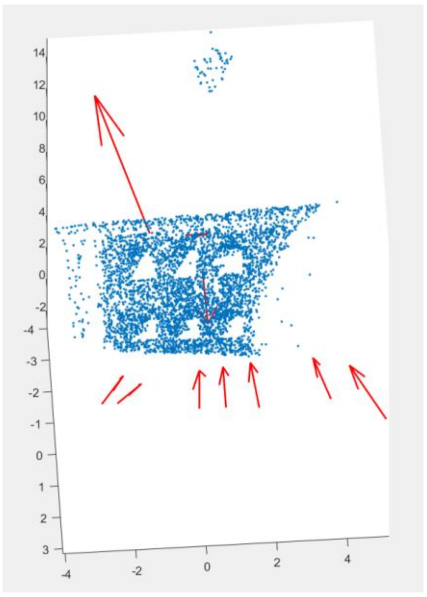


Figure 6.2.1: All points and all cameras after robustly estimation of cameras (Dataset 6.2).

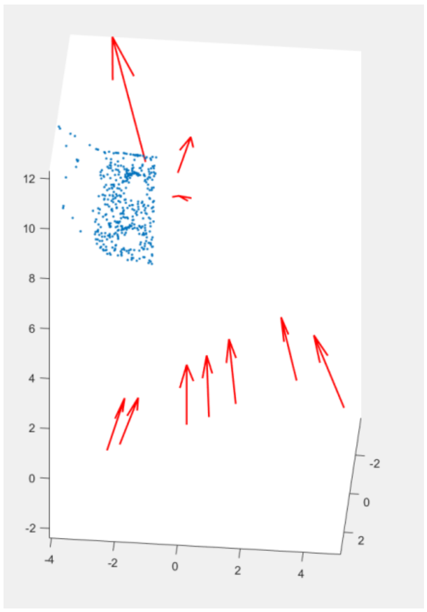


Figure 6.2.2: Partial points and all cameras before bundle adjustment (Dataset 6.2).

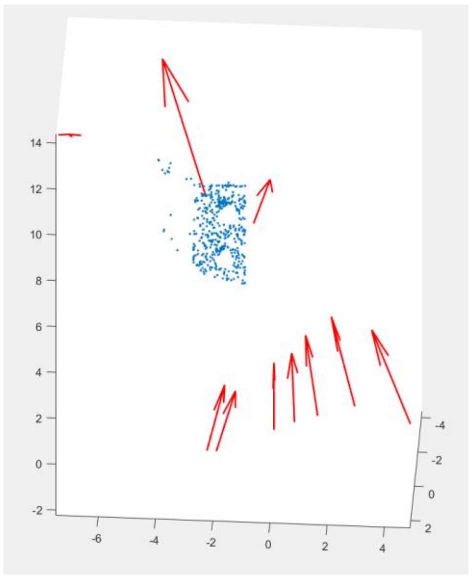


Figure 6.2.3: Partial points and all cameras after bundle adjustment (Dataset 6.2).

Dataset 7 (Skansen Lejonet):
There are 10 images and 10 cameras. As shown in fig 7.1-7.3, all points and all cameras are shown in fig 7.1. Then I select partial points for faster bundle adjustment comparison. Before and after BA can be seen in fig 7.2-7.3. Points and cameras are moved for short distances.

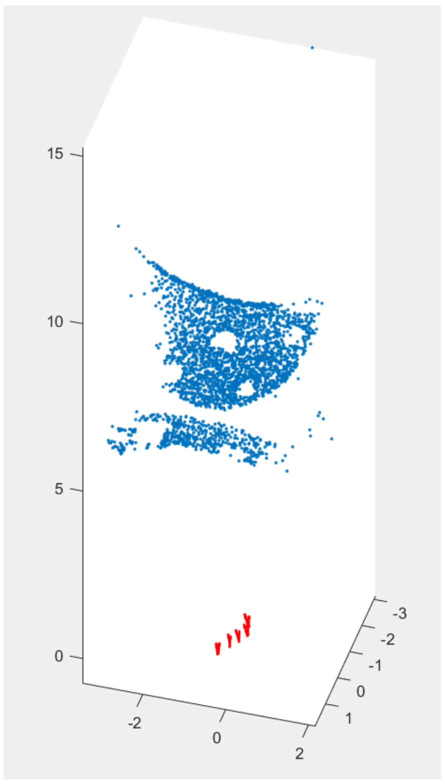


Figure 7.1: All points and all cameras after robustly estimation of cameras (Dataset 7).

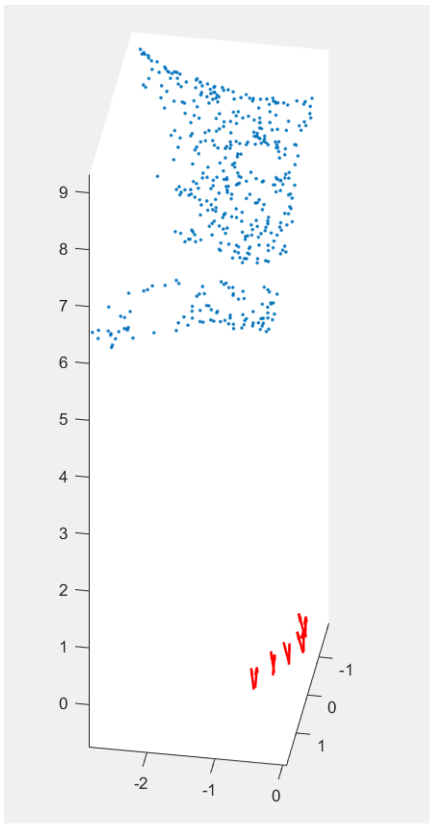


Figure 7.2: Partial points and all cameras before bundle adjustment (Dataset 7).

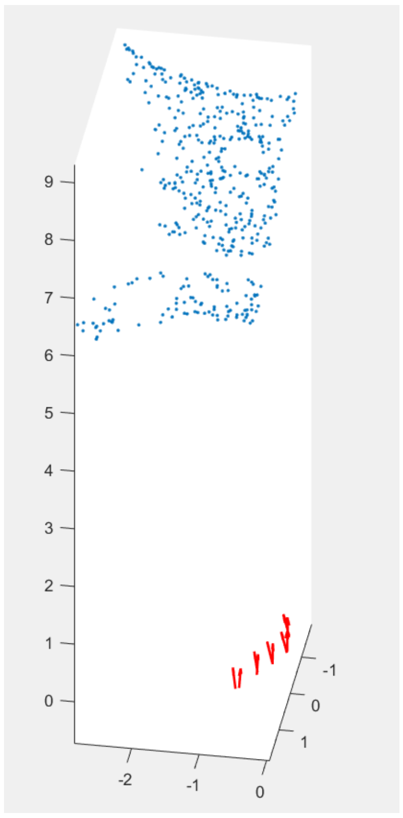


Figure 7.3: Partial points and all cameras after bundle adjustment (Dataset 7).

I also try to take photos by myself, such as below, I took photos of a computer from different angles, but the results are horrible, as seen in fig 8.1-8.2. I try to adjust different combinations of initial pair for long time, but I cannot achieve a good result.

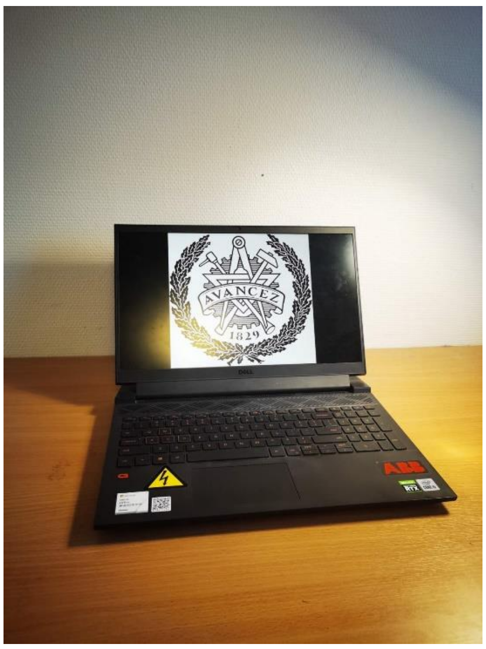


Fig 8.1: Photo of a computer

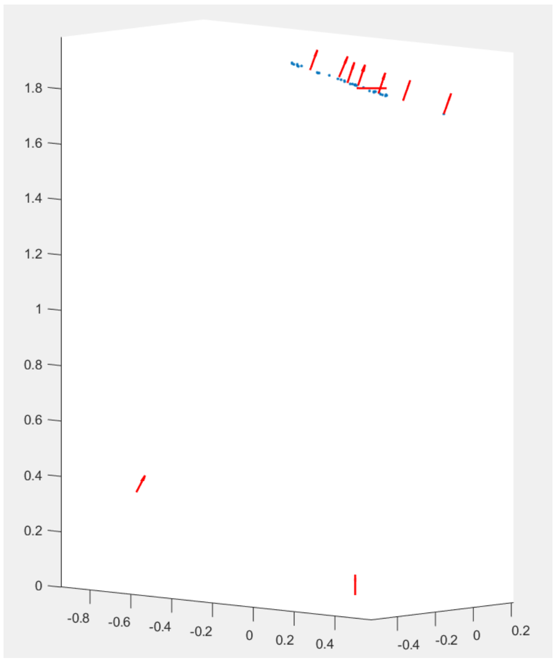


Fig 8.2: 3D reconstructions and cameras

There are lots of good examples from that link, but I am not able to achieve good 3D reconstructions for some examples, so I didn't use them. But from the codes, you can see I try up to 19 different datasets with different details.