EEN020 Computer Vision
Project Report

Vasiliki Kostara
kostara@student.chalmers.se

Chalmers University of Technology
Complex Adaptive Systems

January 2024

Contents

1 Introduction

In this project we implement a 3D reconstruction software for nine distinct image datasets. Firstly, we explain the primary function `run_sfm`, which is run in the `main` and then we will elaborate on the task-specific functions implemented for this project. Explanations for functions created for former assignments and functions implemented for simple calculations, such as `skew`, will be omitted.

The code for this project was written in full collaboration with Alberto Gosalvez White.

2 Functions

2.1 Structure from Motion

The function `run_sfm` contains the principal implementation of this project. During initialization, we retrieve information on a specific dataset and set the respective thresholds. In order to extract features of each image i in the chosen dataset, we use the VLFeat library [1].

The computed SIFT points are used to robustly estimate the essential matrices and the homographies in parallel with the function `estimate_T_and_H_robust` 2.2, through pairs of consecutive images. These are used to extract the relative rotations and translations between each image pair, i.e. the relative orientation $P_{i,i+1} = [R_{i,i+1}\ \mathbf{T}_{i,i+1}]$. The relative orientation is determined after performing a cheirality check for the possible orientations resulting from the essential matrix E with the function `Triangulate3DPoints_CheiralityCheck` 2.3.

The absolute orientations $P_i = [R_i\ \mathbf{T}_i]$ of each image are updated by setting the first camera matrix to $P_1 = [\mathcal{I}\ \mathbf{0}]$ and applying each relative camera matrix to the homography:

$$H_i = \begin{pmatrix} R_i & \mathbf{T}_i \\ \mathbf{0}^T & 1 \end{pmatrix}$$

as follows:

$$P_i = P_{i,i+1} H_i$$

To estimate the translation vectors robustly, we extract features from an initial pair of images determined during initialization. We perform a similar estimation of E and the relative orientation between the initial pair by setting the first camera to $P_{1,init} = [\mathcal{I}\ \mathbf{0}]$. Eventually, we choose the triangulated inliers X that lie mostly in front of both cameras. We compute the descriptors `desc_X` from the matches of the initial image features. We choose one of the two descriptors by comparing the matches given by the descriptor of each image `d_i` and `desc_X`. From these correspondences we estimate the translation vectors robustly and optimize them further with the Levenberg-Marquardt method with the functions `estimate_T_robust` 2.4 and `refine_T_LM` 2.5, contained within the former.

Afterwards, we triangulate the camera matrices and the 2D SIFT points in pairs to obtain the 3D points $\mathbf{X}$ and filter the ones that are very far from the center of gravity $\overline{\mathbf{X}}$, by identifying points $\mathbf{X}_j$ with error $\sqrt{\sum_j \left(\mathbf{X}_j - \overline{\mathbf{X}}\right)^2}$ below the 95^{th} percentile and storing them. Lastly, we visualize the 3D points, along with those extracted from the initial image pair and the camera matrices and present them in Section 3.

2.2 Parallel Essential Matrix and Homography Estimation with RANSAC

In the function `estimate_E_and_H_robust` we compute the essential matrix and the homography with a RANSAC algorithm in parallel. For E we use an 8-point model and for H we use a 4-point model, as shown in Section 2.2.1. During each iterative process, we compute a new E and H and compare the number of inliers each yields compared to the previous E and H. With the addition of H, we use the output of the given function `homog_to_RT` to compute two additional essential matrices $E_1 = [\mathbf{t}_1]_\times R_1$ and $E_2 = [\mathbf{t}_2]_\times R_2$ and find their respective epipolar errors. Consequently, we choose the solution that yields the greatest number of inliers and use it to further estimate $R_{i,i+1}$ and $\mathbf{X}_i$ as described in Section 2.1.

2.2.1 Homography Estimation

In `estimate_homography_DLT` we estimate the homography between two images Im_1 and Im_2. The input arguements of this function are point correspondences between the two images such that $\mathbf{x_2} = H\mathbf{x_1}$, where $\mathbf{x_1} = \begin{bmatrix} x_1 & y_1 & 1 \end{bmatrix}^T$ and $\mathbf{x_2} = \begin{bmatrix} x_2 & y_2 & 1 \end{bmatrix}^T$, $\mathbf{x_1} \in \mathrm{Im}_1$ and $\mathbf{x_2} \in \mathrm{Im}_2$. The homography is computed by performing SVD with four correspondences on the following matrix for each of the point correspondences j :

$$A = \begin{pmatrix} \vdots \\ A_j \\ \vdots \end{pmatrix} = \begin{pmatrix} \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ x_{1,j} & y_{1,j} & 1 & 0 & 0 & 0 & -x_{2,j}x_{1,j} & -x_{2,j}y_{1,j} & -x_{2,j} \\ 0 & 0 & 0 & x_{1,j} & y_{1,j} & 1 & -y_{2,j}x_{1,j} & -y_{2,j}y_{1,j} & -y_{2,j} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \end{pmatrix}$$

We obtain H by reshaping the last column of V in the SVD of $A = USV^T$.

2.3 3D Point Triangulation and Choosing Camera

The function `Triangulate3DPoints_CheiralityCheck`, which receives the camera matrix $P_1 = [\mathcal{I}\ \mathbf{0}]$ and all possible relative orientation matrices $P_{i,i+1}$ extracted from E

$$\begin{bmatrix} UWV^T & u3 \end{bmatrix}, \begin{bmatrix} UWV^T & -u3 \end{bmatrix}, \begin{bmatrix} UW^TV^T & u3 \end{bmatrix}, \begin{bmatrix} UW^TV^T & -u3 \end{bmatrix}$$

along with the inliers of images i and $i+1$. In this function we firstly triangulate 3D points for each possible $P_{i,i+1}$. In order to find the cameras with the most points in front, we compute the scalar product between the principal axis $\mathbf{n}$ and the each 3D point $\mathbf{X}_j$ shifted by the camera center $\mathbf{C}$ and calculate for how many points it is positive. This shows that the principal axis and the 3d point with respect to the camera center lie in the same half-space and, therefore, the 3D point is in front of the respective camera:

$$\text{If } \mathbf{n}^T(\mathbf{X}_j - \mathbf{C}) > 0, \text{ then } \mathbf{X}_j \text{ is in front of the camera.}$$

The chosen relative orientation and 3D point model are signified by the largest sum of the points in front between P_1 and each $P_{i,i+1}$.

2.4 Robust T Estimation

In the function `estimate_T_robust` we develop a 2-point RANSAC algorithm. The input arguements are the corresponding 2D and 3D matched points as described in Section 2.1, the rotation matrix of each image and a threshold. After the $\mathbf{T}$ is calculated (Section 2.4.1) in each iteration, the inliers are determined by the reprojection error for all matched points j:

$$e = \frac{\sqrt{\sum_j \mathbf{x}_j - \mathbf{proj}_{\mathbf{X_j}}}}{2}$$

where $\mathbf{proj}_{\mathbf{X_j}} = R\mathbf{X}_j + \mathbf{T}$ flattened.

2.4.1 Translation Estimation

The function `estimate_T_DLT` estimates a value for the translation vector, by using two of the corresponding 2D and 3D matched points and the rotation matrix of each image. Let $\mathbf{x_1}$, $\mathbf{x_2}$ be the 2D and $\mathbf{X_1}$, $\mathbf{X_2}$ be the 3D correspondences. For a rotation matrix R, the translation vector $\mathbf{T}$ is the solution to the system:

$$A \cdot \mathbf{T} = B$$

where

$$A = \begin{pmatrix} [\mathbf{x_1}]_\times \\ [\mathbf{x_2}]_\times \end{pmatrix} \quad \text{and} \quad B = \begin{pmatrix} [-\mathbf{x_1}]_\times \\ [-\mathbf{x_2}]_\times \end{pmatrix}$$

For the solution of the above linear system, we code the following operation [2]:

$$\mathbf{T} = A \backslash B$$

2.5 Levenberg-Marquardt

The function `refine_T_LM` is contained in the last steps of function `estimate_T_robust`. We input the robustly estimated $\mathbf{T}$, the corresponding 2D and 3D matched points, the rotation matrix and the inliers yielded by the 2-point algorithm. For a specific number of iterations, we minimize for each image i and point j

$$\sum_i \sum_j \left(x_{ij}^1 - \frac{\mathbf{P}_i^1 \mathbf{X}_j}{\mathbf{P}_i^3 \mathbf{X}_j},\ x_{ij}^2 - \frac{\mathbf{P}_i^2 \mathbf{X}_j}{\mathbf{P}_i^3 \mathbf{X}_j} \right) = \sum_i \sum_j \left(x_{ij}^1 - \frac{\mathbf{R}_i^1 \mathbf{X}_j + T^1}{\mathbf{P}_i^3 \mathbf{X}_j + T^3},\ x_{ij}^2 - \frac{\mathbf{R}_i^2 \mathbf{X}_j + T^2}{\mathbf{R}_i^3 \mathbf{X}_j + T^3} \right)$$

We use the same algorithm as for optimizing the inliers, but with respect to $\mathbf{T} = \begin{bmatrix} T^1 & T^2 & T^3 \end{bmatrix}^T$.

2.5.1 Jacobian

In order to optimize $\sum_i \sum_j \left(x_{ij}^1 - \frac{\mathbf{R}_i^1 \mathbf{X}_j + T^1}{\mathbf{P}_i^3 \mathbf{X}_j + T^3},\ x_{ij}^2 - \frac{\mathbf{R}_i^2 \mathbf{X}_j + T^2}{\mathbf{R}_i^3 \mathbf{X}_j + T^3} \right)$, we calculate the Jacobian with respect to $\mathbf{T}$:

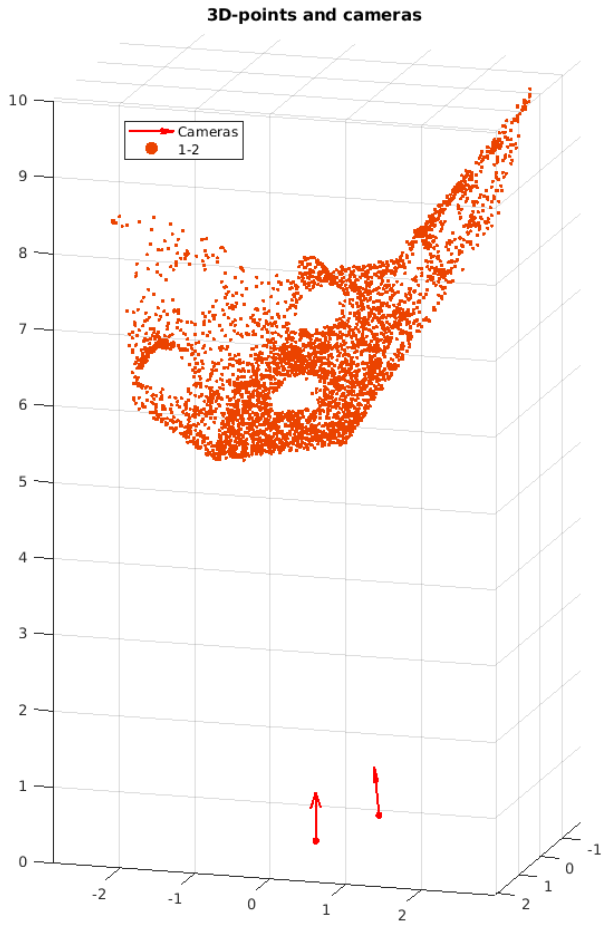
$$J(\mathbf{T}) = \begin{pmatrix} -\frac{1}{\mathbf{P}_i^3 \mathbf{X}_j} & 0 & \frac{\mathbf{P}_i^1 \mathbf{X}_j}{(\mathbf{P}_i^3 \mathbf{X}_j)^2} \\ 0 & -\frac{1}{\mathbf{P}_i^3 \mathbf{X}_j} & \frac{\mathbf{P}_i^2 \mathbf{X}_j}{(\mathbf{P}_i^3 \mathbf{X}_j)^2} \end{pmatrix}$$

and use it in order to update $\mathbf{T}$ as $\mathbf{T} + d\mathbf{T}$.

3 Visualization

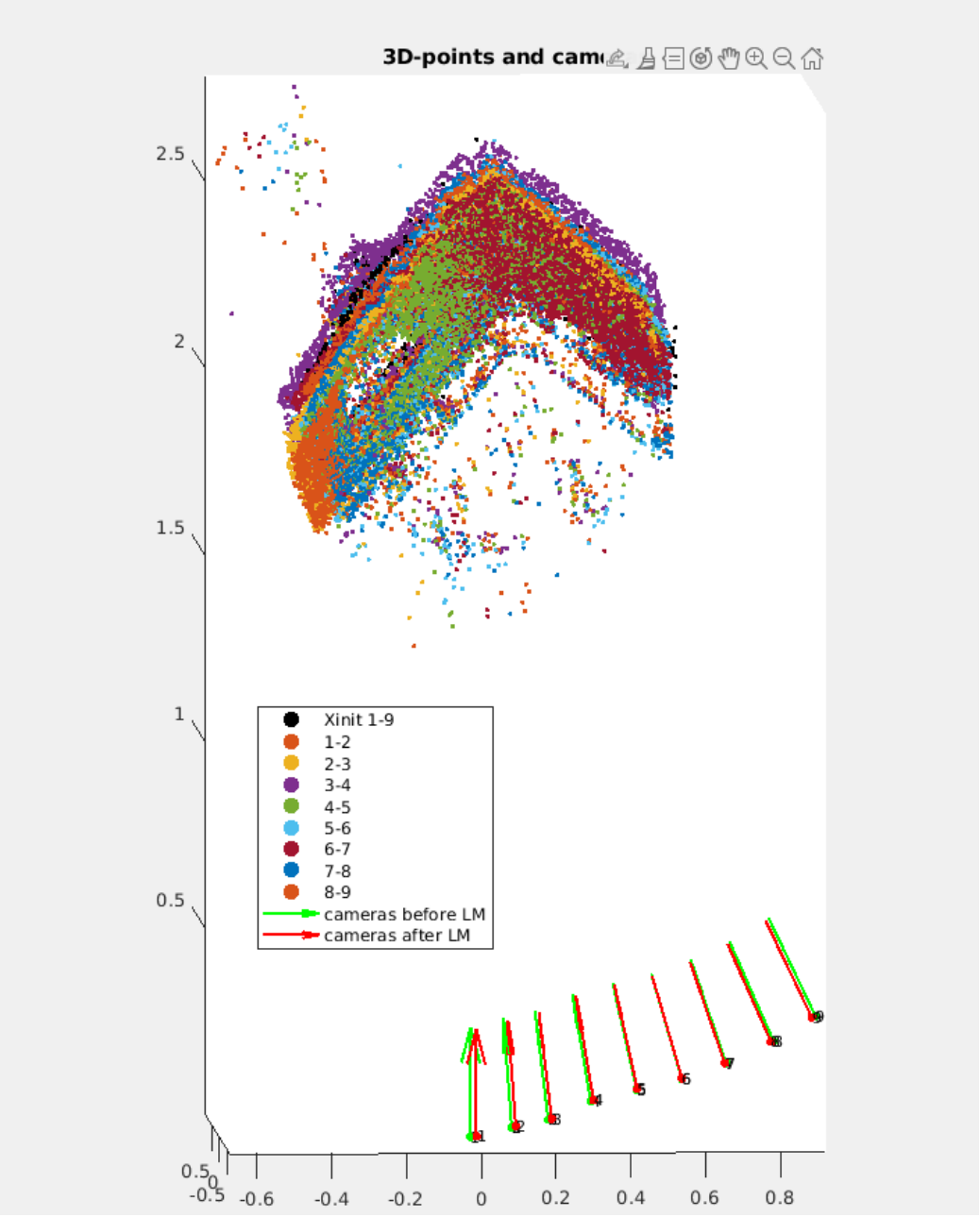
Below are presented the plots of the 3D reconstructions for all the datasets.

3.1 Dataset 1



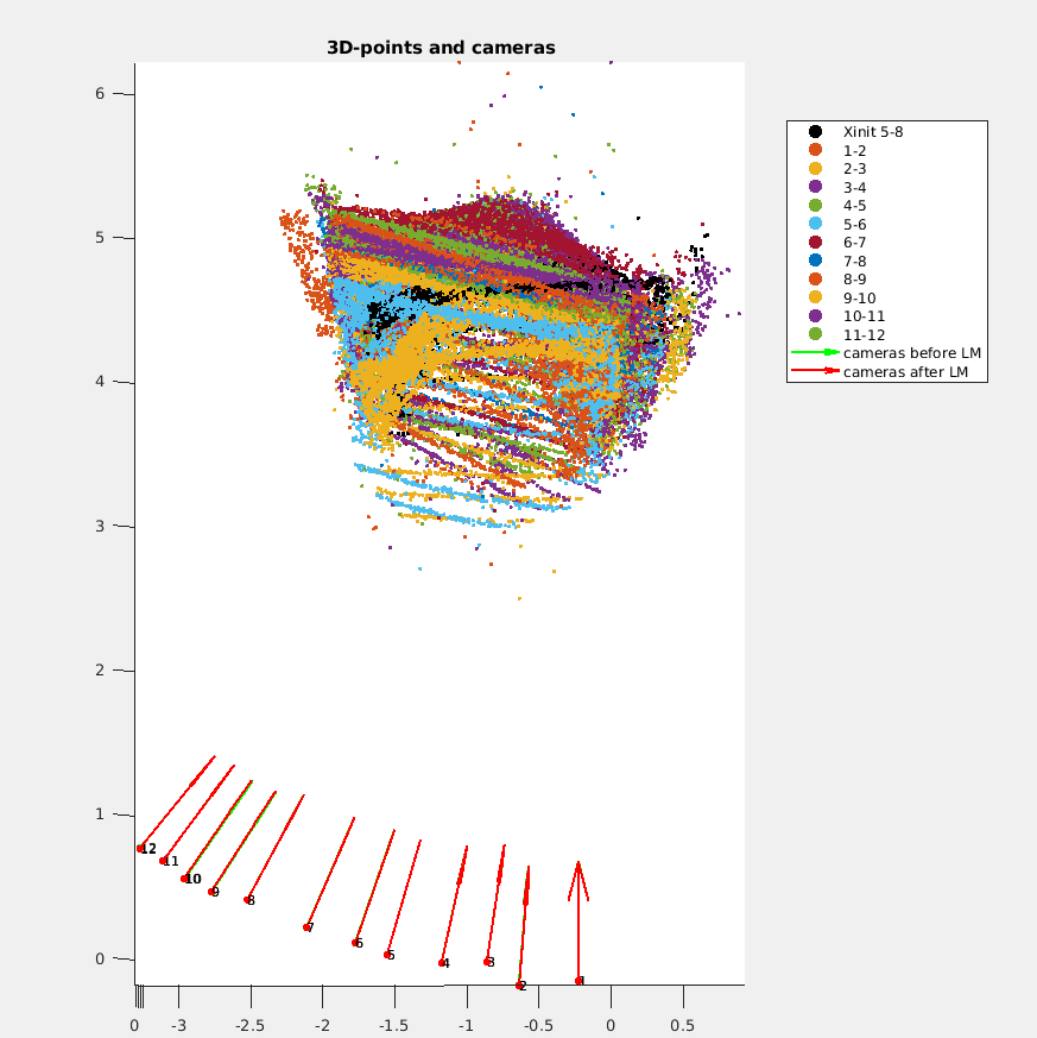
Dataset 1

3.2 Dataset 2



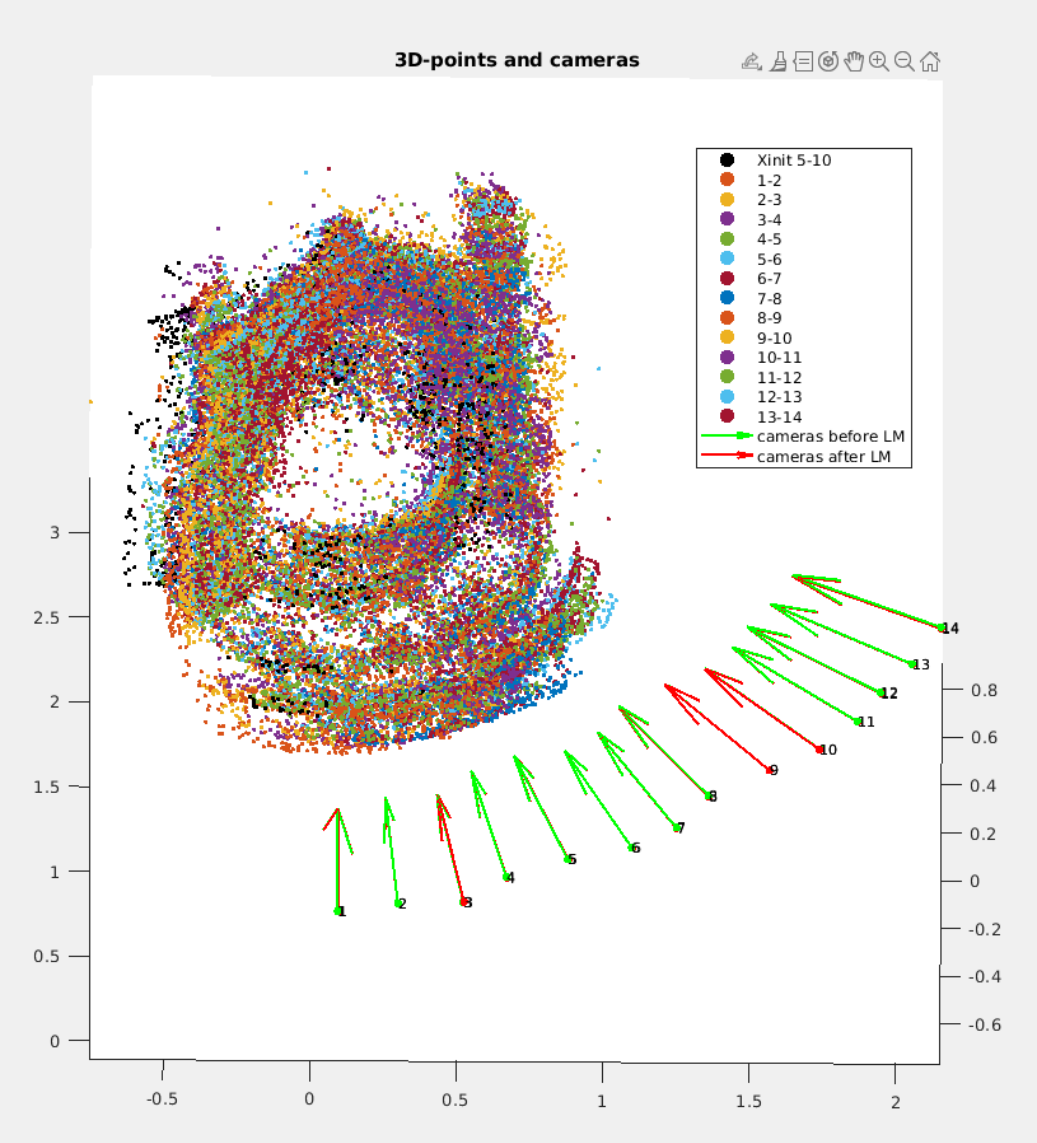
Dataset 2

3.3 Dataset 3



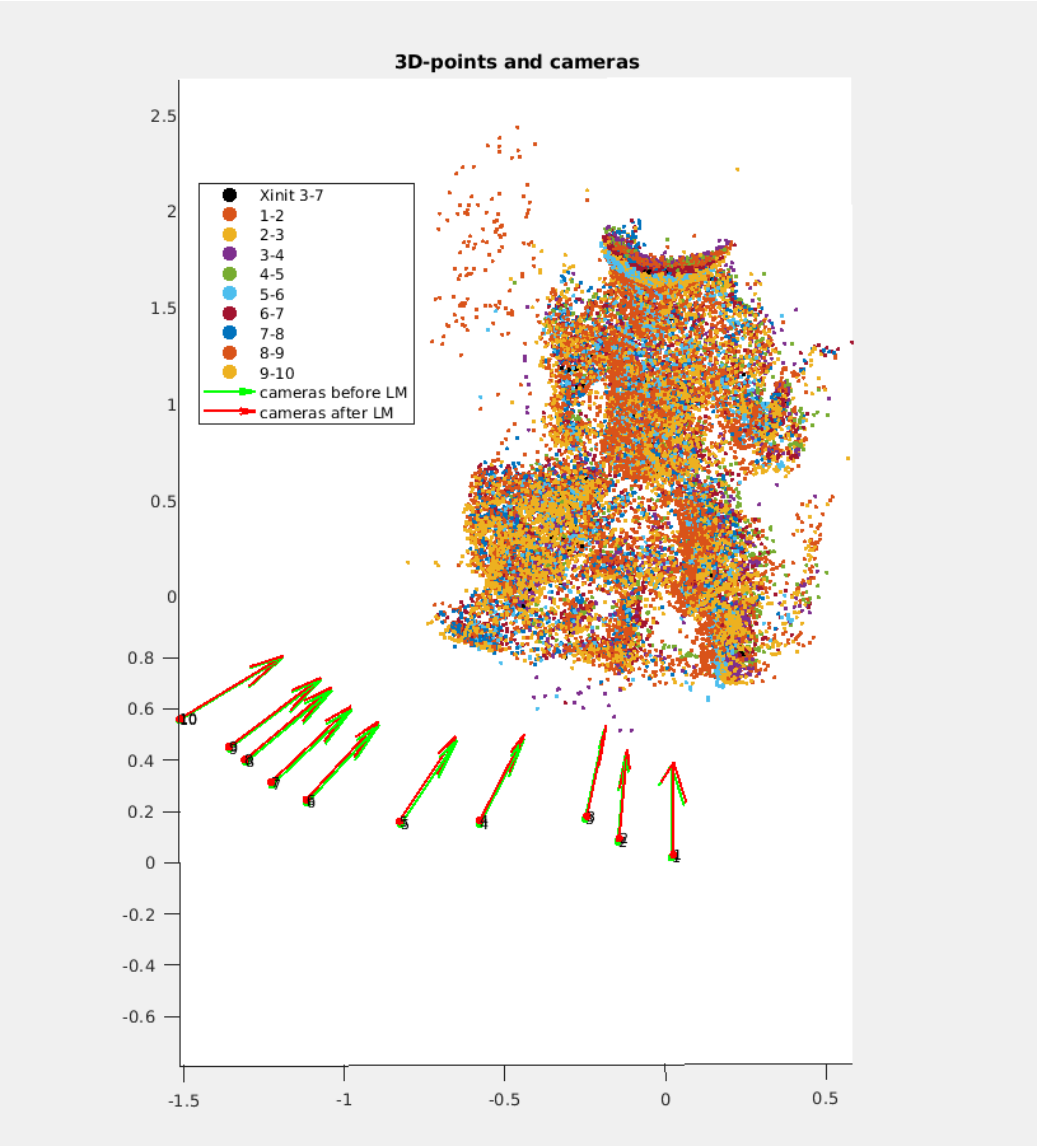
Dataset 3

3.4 Dataset 4



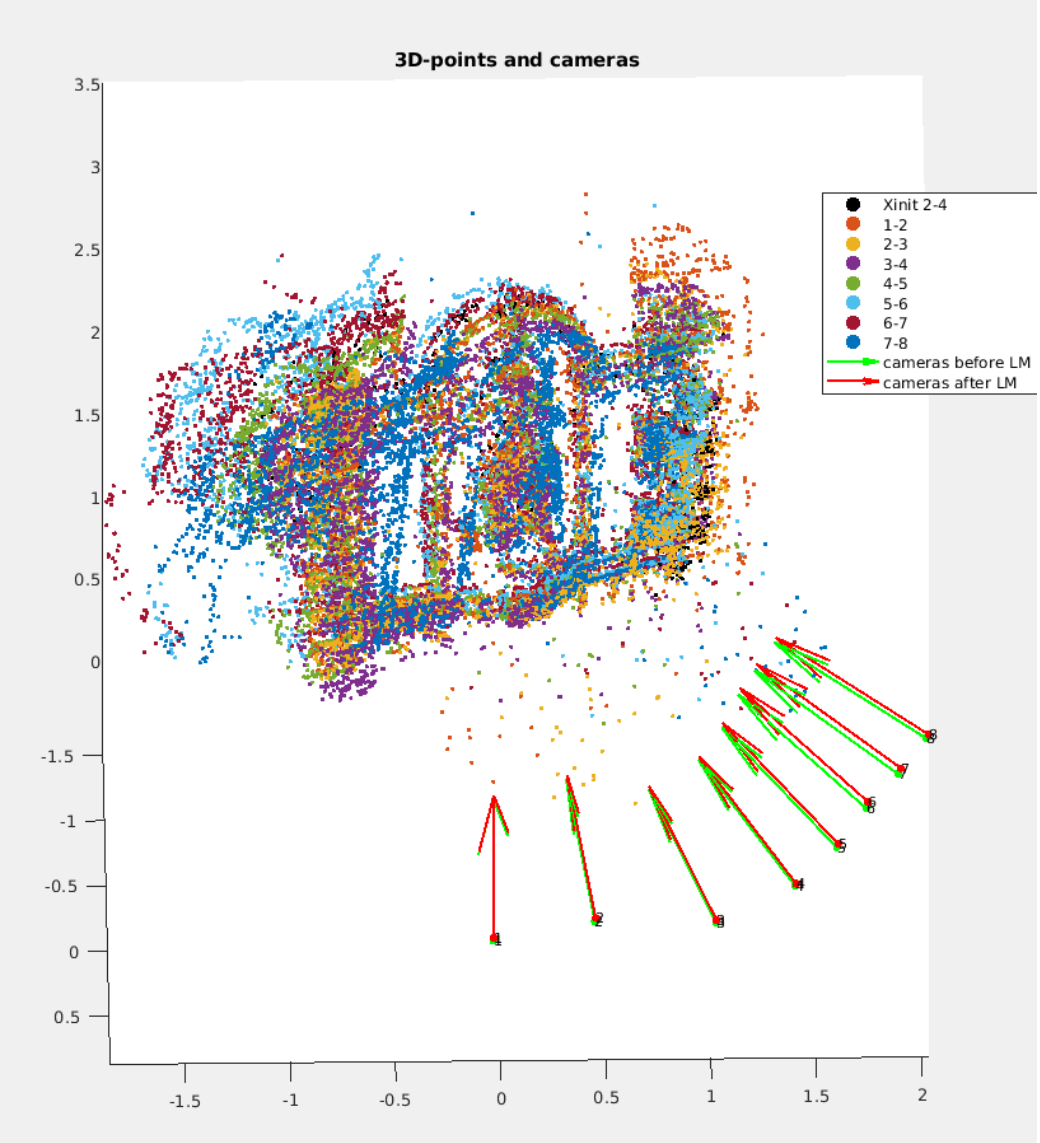
Dataset 4

3.5 Dataset 5



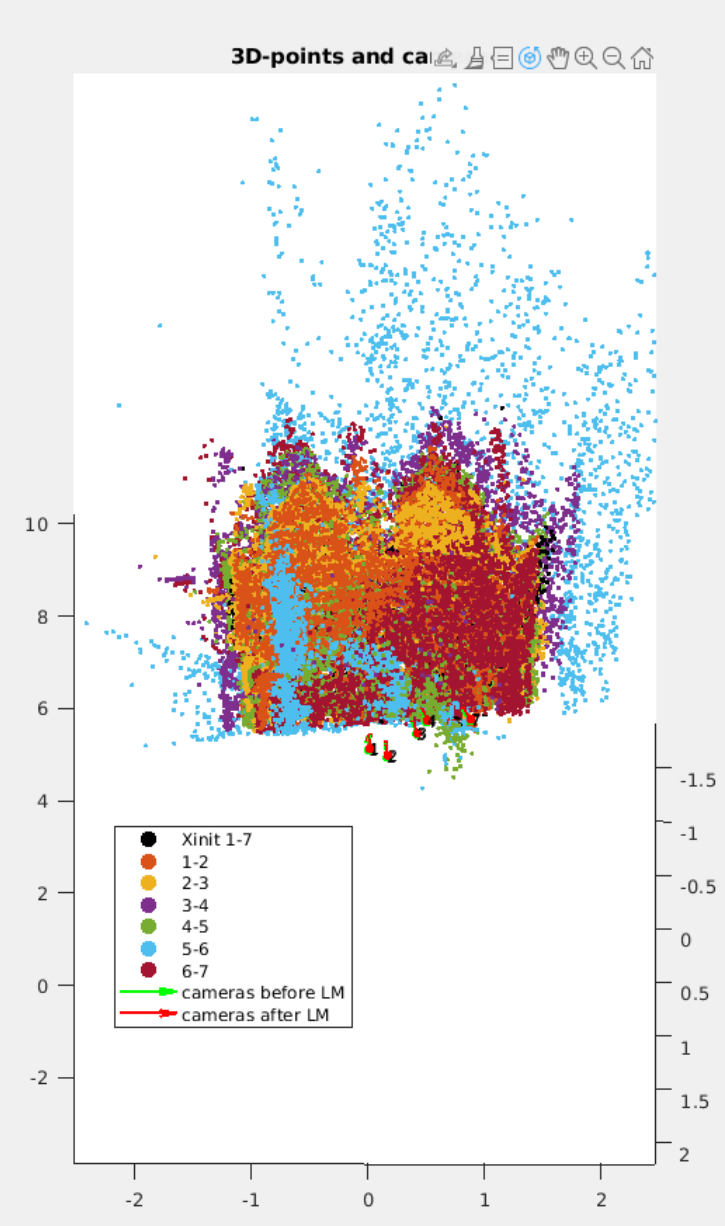
Dataset 5

3.6 Dataset 6



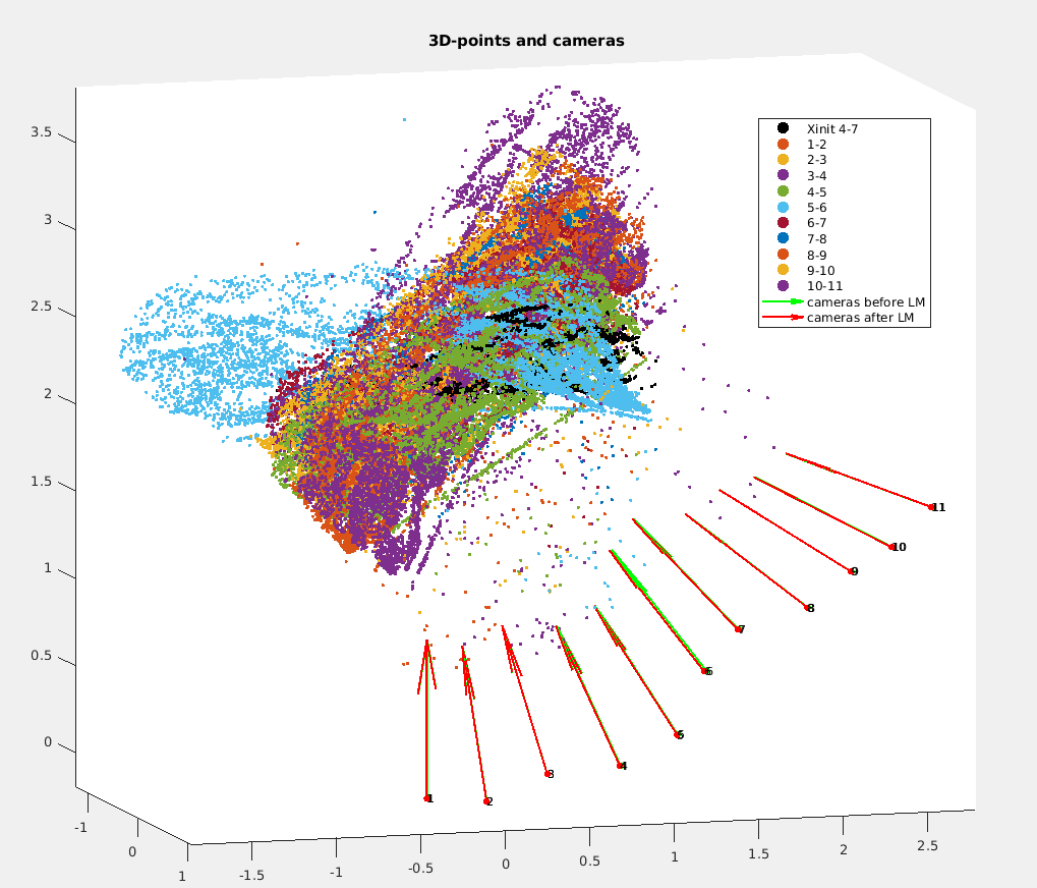
Dataset 6

3.7 Dataset 7



Dataset 7

3.8 Dataset 8



Dataset 8

4 Discussion

After reevaluating the estimation of the homography in Section 2.2.1 and of the translation vector in Section 2.4.1 the results become significantly clearer. The cameras before the Levenberg-Marquardt appear in expected positions and are slightly altered after the optimization. Most of the images show satisfactory results, that remind of the structure seen in the corresponding dataset. The only exceptions are the datasets 8 and 9 (omitted), the reconstructions of which do not meet the desired criteria. As suggested during the course lectures, planar datasets like these would require different algorithms to be successfully reconstructed. During the project development period, we tried to implement the 5-point algorithm, but without success. As an outcome, it is important to acknowledge the time limitations of this project, despite our enthusiasm to explore its full potential.

References

[1] A. Vedaldi and B. Fulkerson, "VLFeat: An open and portable library of computer vision algorithms," http://www.vlfeat.org/, 2008.

[2] MathWorks. (2022) mldivide function documentation. MATLAB Documentation. [Online]. Available: https://se.mathworks.com/help/matlab/ref/mldivide.html

[3] T. Opsahl. (2019) Lecture 4.3: Estimating homographies from feature correspondences. TEK5030–Computer Vision. University of Oslo. [Online]. Available: https://www.uio.no/studier/emner/matnat/its/TEK5030/v19/lect/lecture-4-3-estimating-homographies-from-feature-correspondences.pdf