Project EEN020 Computer Vision

Erik Norlin

4 January 2024

Algorithm

The algorithm consists of four parts; rotation averaging to estimate the absolute rotations of the cameras, translation registration to estimate the translations of the cameras, camera refinement to bundle adjust the cameras, and triangulation of the final 3D-reconstruction.

Rotation averaging

All absolute rotations are obtained by chaining the relative rotations of each adjacent camera pair as $R_j = R_{i,j}R_i$. The relative rotations are extracted from robustly estimated essential matrices and normalized by taking the singular value decomposition of $R_{i,j}$ and setting $R_{i,j} = UV^T$.

Essential matrices for each adjacent camera pair are robustly estimated using image correspondences obtained from SIFT-points in a RANSAC loop. `estimate_E_robust` from assignment 4 is extended by also extracting essential matrices from an estimated homography, checking the validity of every estimated essential matrix, and dynamically changing the number of RANSAC iterations T_E and T_H by computing

$$T = \left\lceil \frac{\ln(1-\alpha)}{\ln(1-\epsilon^s)} \right\rceil \tag{1}$$

Since this formula can be aggressive in the sense that the number of RANSAC iterations can be very large, and sometimes be very small, they are therefore capped at a maximum and minimum number of iterations. The number of RANSAC iterations can also be scaled by a factor.

In each iteration both an essential matrix and a homogropahy are estimated for minimal samples using an 8-point DLT method and a 4-point DLT method respectively. From the homography, two other essential matrices can be extracted. The reason for extracting essential matrices from a homography is that as the scene becomes more planar the risk of degenerate solutions obtained by the 8-point method increases, whereas a homography becomes better at mapping the image points of two cameras facing the scene. Inliers for

an essential matrix are measured with point-to-line distance (point to epipolar line), and inliers for a homography are measured with point-to-point distance (point to projected point). The pixel-threshold is set to be three times larger for computing inliers for the homography than the essential matrices. A valid essential matrix has rank 2 and is therefore always tested for this. If a new best essential matrix is estimated but is not rank 2 then it is discarded. T_E and T_H are adjusted every time a better valid essential matrix is found.

When RANSAC has terminated, four cameras are extracted from the best essential matrix *outside* the RANSAC loop to speed up the process. The cheirality test is perfomed on these cameras, and the camera that yields the most inliers in front of both cameras is selected as the correct solution. For the vast majority of the time this works wonderful. However in rare cases, triangulating 3D-points using the correct extracted camera from the returned valid essential matrix results in non-sensical "sprays" of point clouds.

Translation registration

An initial 3D-reconstruction is triangulated with relative cameras and image correspondences (SIFT-matches) from an initial camera pair with a sufficiently large baseline. Outliers are removed as well as 10% of the 3D-points being the furthest away from the center of gravity of the point cloud. The initial 3D-reconstruction is not rotated nor centered because the canonical camera in the common coordinate frame is set to be camera P_i of the initial pair. Hence, rotation and centering of the 3D-points are not needed.

2D-3D correspondences are established for each camera by matching descriptors corresponding to the 3D-points and the descriptors for the image points obtained from SIFT.

Camera translations are robustly estimated in `estimate_T_robust` in a RANSAC loop one by one for a minimal sample of 2 by solving $\mathbf{T}_i$ in eq. 2

$$\Big([\mathbf{x}_{ij}]_\times \mid [\mathbf{x}_{ij}]_\times R_i\Big) \begin{pmatrix} \mathbf{T}_i \\ \mathbf{X}_j \end{pmatrix} = \mathbf{0} \tag{2}$$

3D points are projected onto the image plane using the absolute rotation and an estimate of the translation vector, and the 2D-3D inliers are measured using point-to-point distance between the projected points and the corresponding image points.

Initially, the translations were estimated using a 2-point method obtained by simplifying `estimate_P_DLT` from assignment 2. Unfortunately, this method did not result in inliers for all cameras, only for some, even with extremely large values for the pixel-threshold. At times, some cameras only get one inlier, and these initial estimates turns out to never be good enough for the bundle adjustment to yield satisfactory final 3D-reconstructions. Instead, solving $\mathbf{T}_i$ from eq. 2 for a minimal sample of two showed to result in more inliers and better 3D-reconstructions after bundle adjustment. Enough inliers for all translations are still not obtained by doing this, though. On top of this, when triangulating 3D-reconstructions using poor robust estimates of the translations, optimized or not, the SVD can sometime have a hard time converging resulting in throwing an exception. With all these issues at hand, non-robust translations are also estimated using all 2D-3D correspondences as a compromise to get translation vectors for all cameras.

Camera refinement

The absolute rotations and translations are refined by minimizing the squared reprojection error using Levenberg-Marquardt (LM) where the rotations are parametrized as quaternions when being optimized.

A modified version of LM from assignment 4 was first implemented for this where the rotations and the translations are optimized and the 3D-points assumed fixed. The rotations were first parametrized to axis-angle representation but doing this was infeasible. The rotations of the cameras are so small for some data sets that the angle representation is practically zero resulting in singularity in the parametrization. Instead, the rotations are parametrized as quaternions which avoids this issue.

The updates $\delta\mathbf{T}_i$ and $\delta\mathbf{q}_i$ are computed using the jacobian of the residuals. The jacobian of the residual with respect to the translation parameters are analytically calculated as

$$\frac{\partial \mathbf{r}}{\partial \mathbf{T}_i^1} = \begin{bmatrix} \frac{-1}{R_i^3\mathbf{X}_j+\mathbf{T}_i^3} \\ \mathbf{0} \end{bmatrix}, \frac{\partial \mathbf{r}}{\partial \mathbf{T}_i^2} = \begin{bmatrix} \mathbf{0} \\ \frac{-1}{R_i^3\mathbf{X}_j+\mathbf{T}_i^3} \end{bmatrix}, \frac{\partial \mathbf{r}}{\partial \mathbf{T}_i^3} = \begin{bmatrix} \frac{R_i^1\mathbf{X}_j+\mathbf{T}_i^1}{(R_i^3\mathbf{X}_j+\mathbf{T}_i^3)^2} \\ \\ \frac{R_i^2\mathbf{X}_j+\mathbf{T}_i^2}{(R_i^3\mathbf{X}_j+\mathbf{T}_i^3)^2} \end{bmatrix} \tag{3}$$

The jacobian of the residual with respect to the quaternion parameters are tedious but are implemented and can be found in `compute_jacobian_of_residual_wrt_q` in `computer_vision.py`.

The cameras are optimized one by one and LA is terminated when the reprojection error is "too close" to the reprojection error of the previous iteration. However, the cameras that are returned from the optimization are essentially the same as before. This is especially true for the robustly estimated translations. As mentioned before, not many 2D-3D inliers are obtained from `estimate_T_robust`. This leads to minuscule reprojection errors and hence instantaneous convergence even though the refined cameras yield disastrous reconstructions. The non-robust translations result in larger reprojection error and LA is capable of decreasing this, however only a tiny fraction. One fix that could potentially improve the implementation is to optimize over all parameters of all cameras simultaneously in order to increase the reprojection error and give LA more to work with, so to speak. It still remains unclear why this implementation does not work as expected, though. For further details about this implementation see `optimize_T_and_R` in `computer_vision.py`.

Resorting to other options, LM is instead implemented using Scipy's least squares module where LM is an available method. Refining all cameras at once using this module is convenient and it turns out that this yields satisfying 3D-reconstructions. The 3D-reconstructions from each adjacent camera pair aligns closely with each other with slight offset, which is reasonable since this is not a full scale bundle adjustment.

Final 3D-reconstruction

Two final 3D-reconstructions are obtained by accumulating triangulated 3D-points using cameras and image correspondences for each adjacent camera pair. The first showing the reconstruction of the un-refined camera pairs, and the other showing the reconstruction of the refined camera pairs.

Running the software

The modules that needs to be installed to run this software are

- `argparse`
- `matplotlib`
- `numpy`
- `opencv-python`
- `scipy`
- `tqdm`

The files of the actual implementation are

- `computer_vision.py`
- `get_dataset_info.py`
- `main.py`
- `pipeline.py`

which all need to be in the working directory together with a `data` folder containing numerated sub-folders with data sets. `computer_vision.py` contains more general functions for computer visions tasks such as robust estimations and LM, and `pipeline.py` contains larger functions for the specific steps of the project such as rotation averaging, translation registration etc.

Run the software with `main.py -dataset <dataset> -T_robust`. `-dataset` is required and takes an integer and searches for the specified data set in the `data` folder. `-T_robust` is a flag and is not required, and if passed in, the translation vectors will be estimated robustly, but the convergence of SVD can occasionally fail when triangulating the final 3D-reconstruction due to poor estimates of the robust translations. If left out, all 2D-3D correspondences will be used to estimate the translation vectors. Leaving `-T_robust` out yields significantly better 3D-reconstructions after bundle adjustment in this implementation.

Reconstruction of the data sets with `T_robust=True`

Figures 1-14 show reconstructions of the data sets before and after LM optimization for robust translations.

In data sets 3-7, the pixel-threshold for estimation of the essential matrix was the threshold of recommendation, and the threshold for estimation of the homography was 3 times the recommended threshold. For data sets 8 and 9 the recommended threshold was increased three times when estimating the initial essential matrix. For data sets 3-8, the pixel-threshold for robustly estimating the translations was set to 10 times larger than the recommended pixel-threshold in order to get 2D-3D inliers for most cameras. For data set 9, this threshold was increased to 20.

The 3D-point clouds obtained from triangulation from all adjacent camera pairs have different colors to visually differentiate them. The reconstructions does not look great before optimization, and afterwards we can see that most often a couple of point clouds

align with each other with some offset showing that LM has improved the solutions. The reconstruction of data set 7 after LM in Figure 10 turns out to be the only data set where all translations are successfully robustly estimated and yields a proper 3D-reconstruction. The most misaligned point clouds were the ones triangulated using cameras with poorly estimated translations (very few inliers).

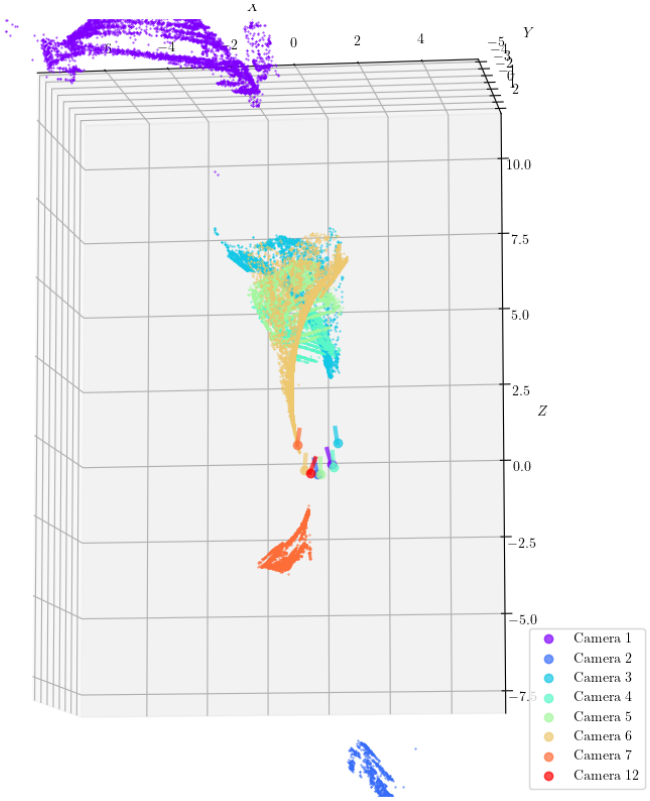

Figure 1: Reconstruction of data set 3 before LM.

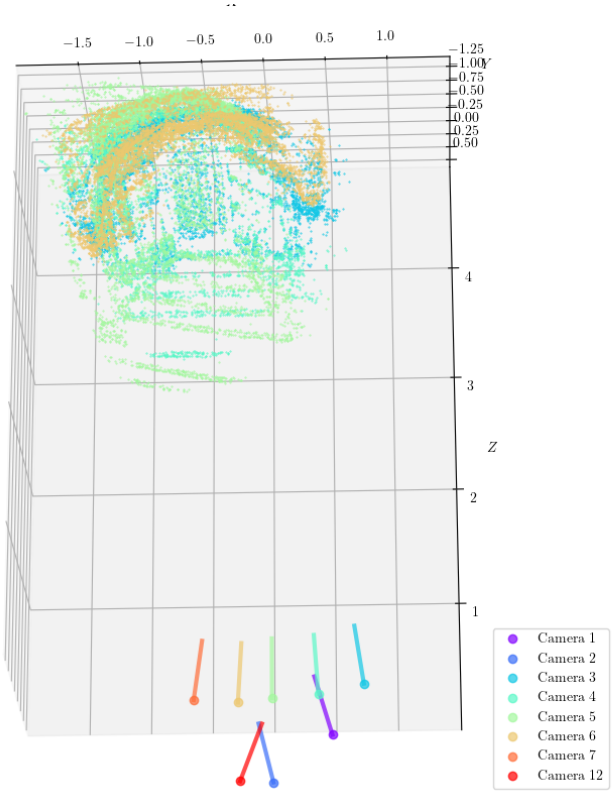

Figure 2: Reconstruction of data set 3 after LM. 4 point clouds align out of 11 possible.

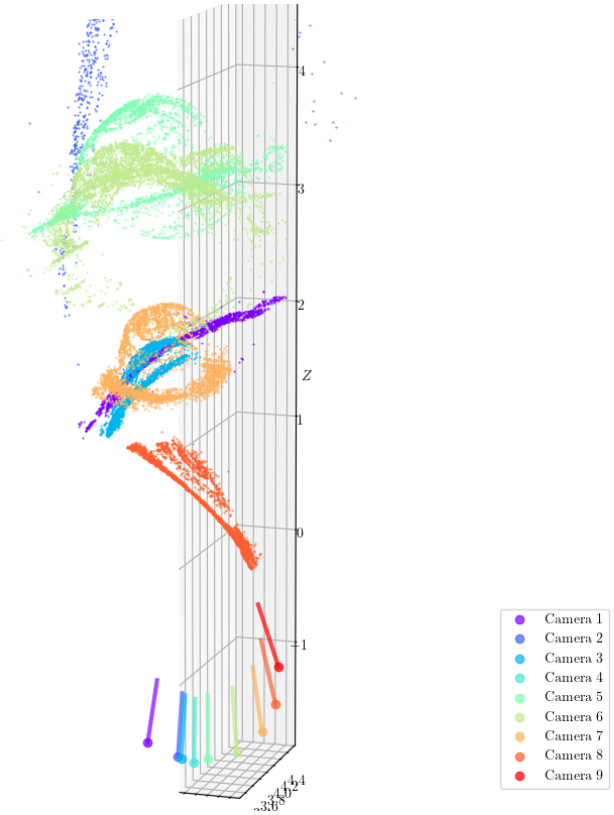

Figure 3: Reconstruction of data set 4 before LM.

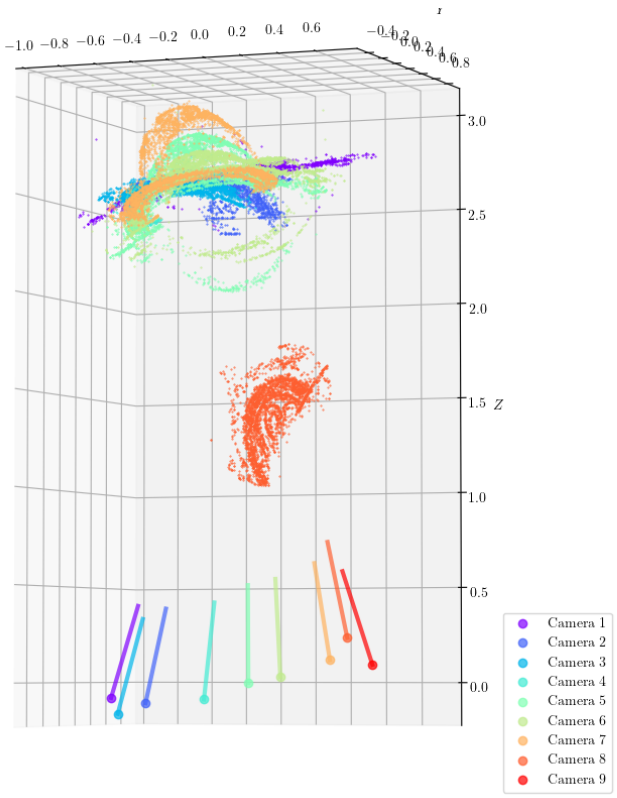

Figure 4: Reconstruction of data set 4 after LM. 3 point clouds align reasonably well out of 13 possible.

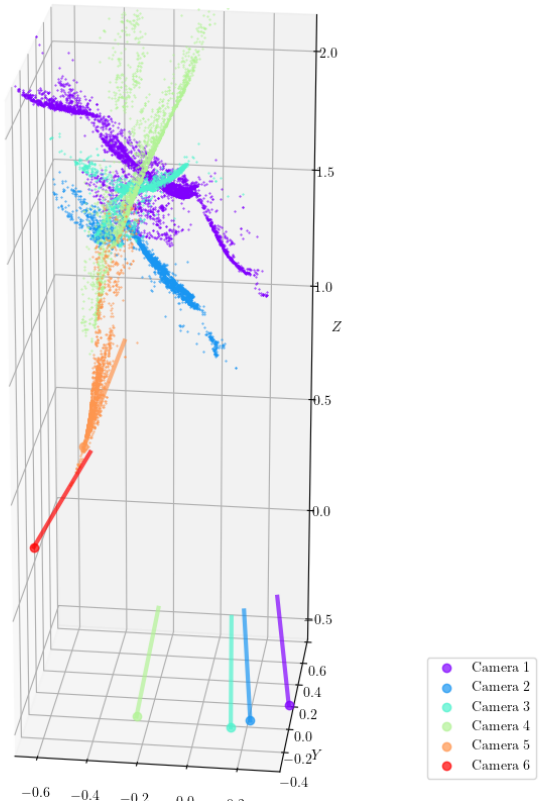

Figure 5: Reconstruction of data set 5 before LM.

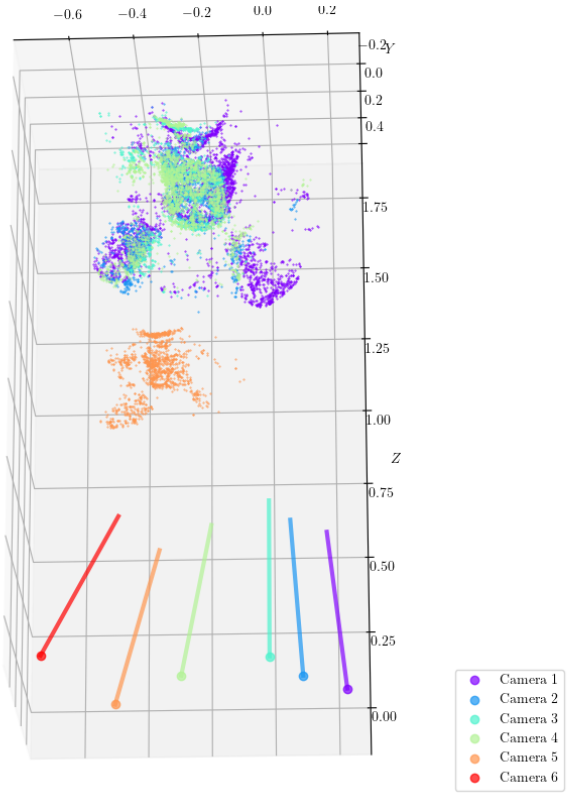

Figure 6: Reconstruction of data set 5 after LM. 4 point clouds align out of 9 possible.

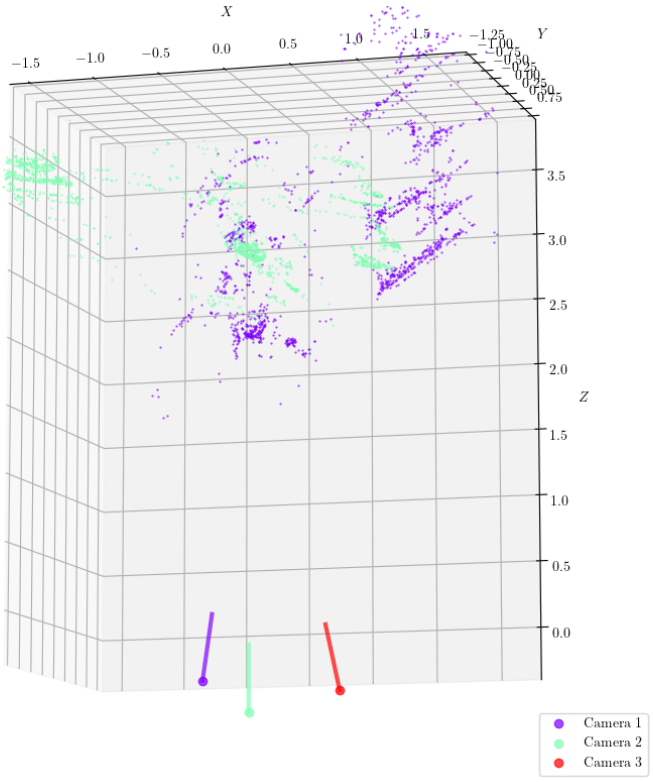

Figure 7: Reconstruction of data set 6 before LM.

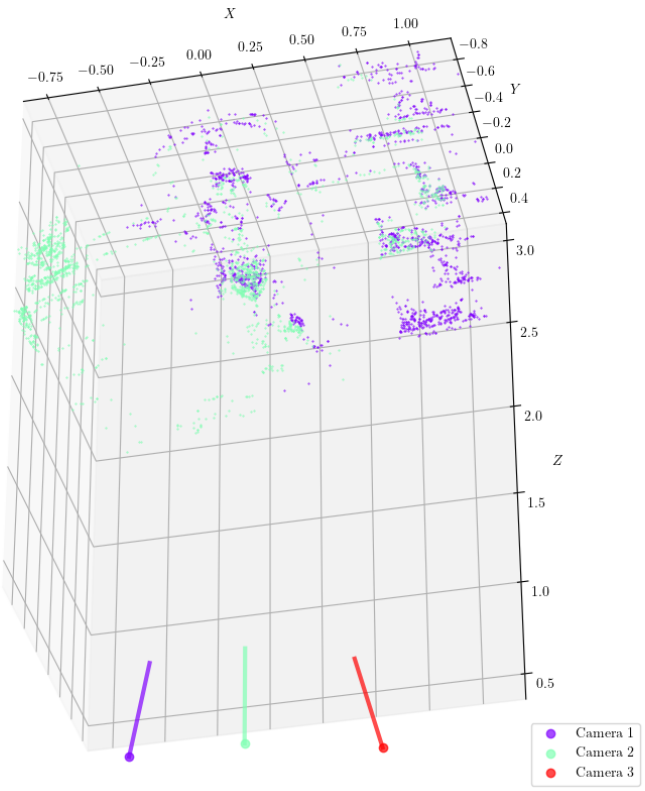

Figure 8: Reconstruction of data set 6 after LM. 2 point clouds align out of 7 possible.

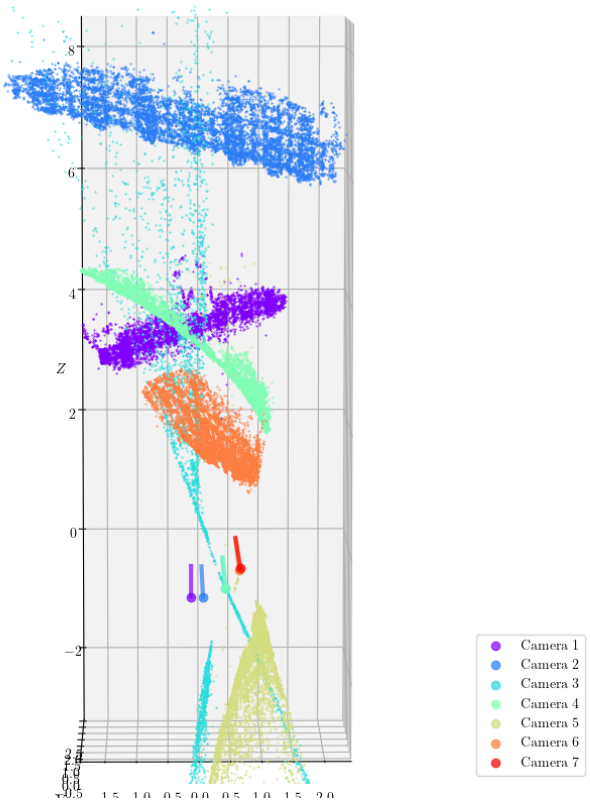

Figure 9: Reconstruction of data set 7 before LM.

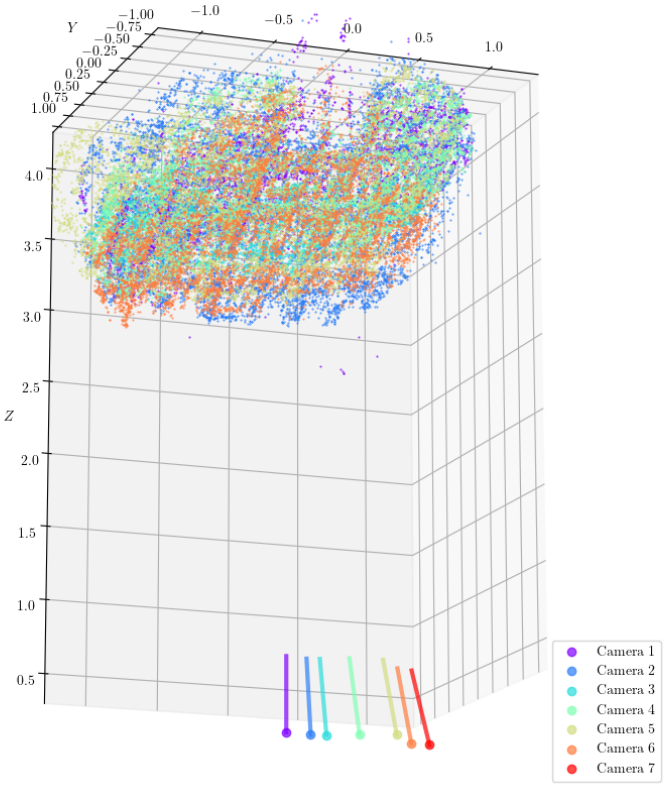

Figure 10: Reconstruction of data set 7 after LM. This is the only data set where all translations were successfully robustly estimated and resulted in a reasonable final reconstruction. 6 point clouds align reasonably well out of 6 possible.

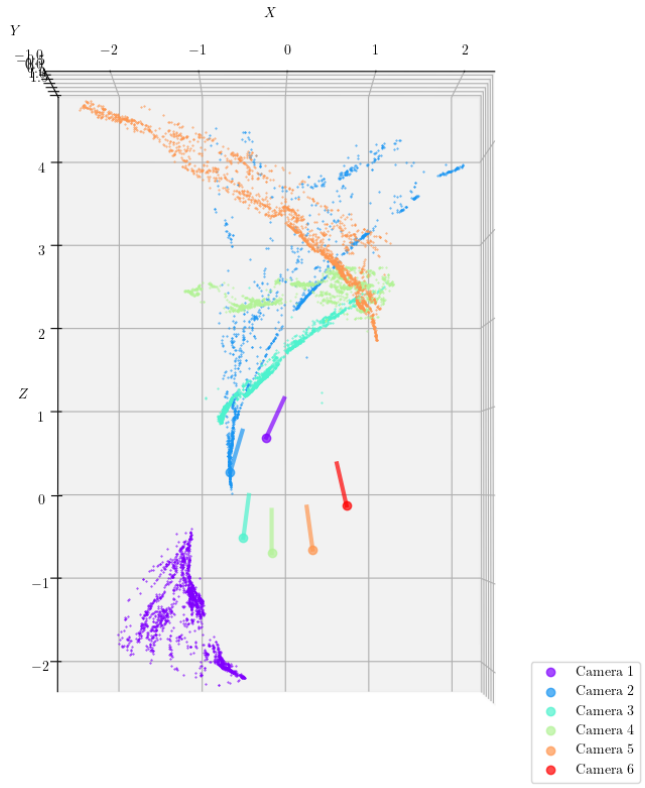

Figure 11: Reconstruction of data set 8 before LM.

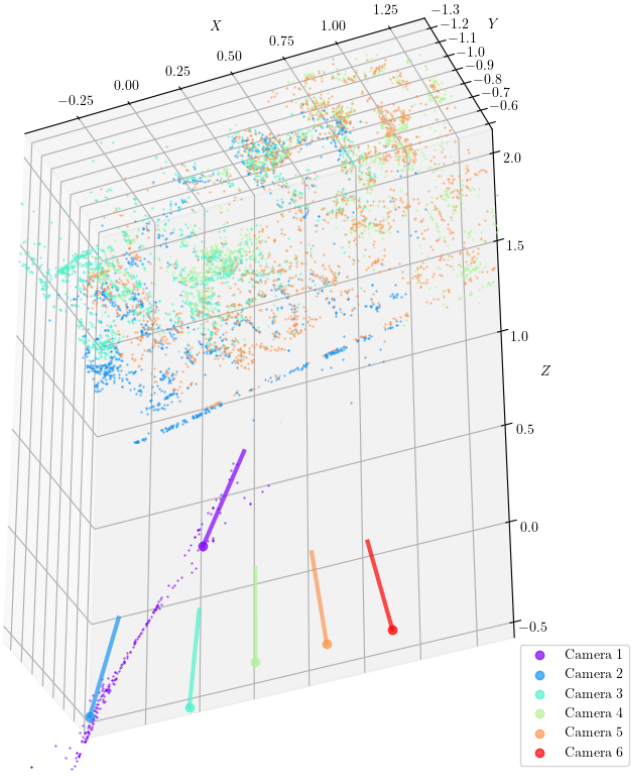

Figure 12: Reconstruction of data set 8 after LM. 4 point clouds align out of 11 possible.

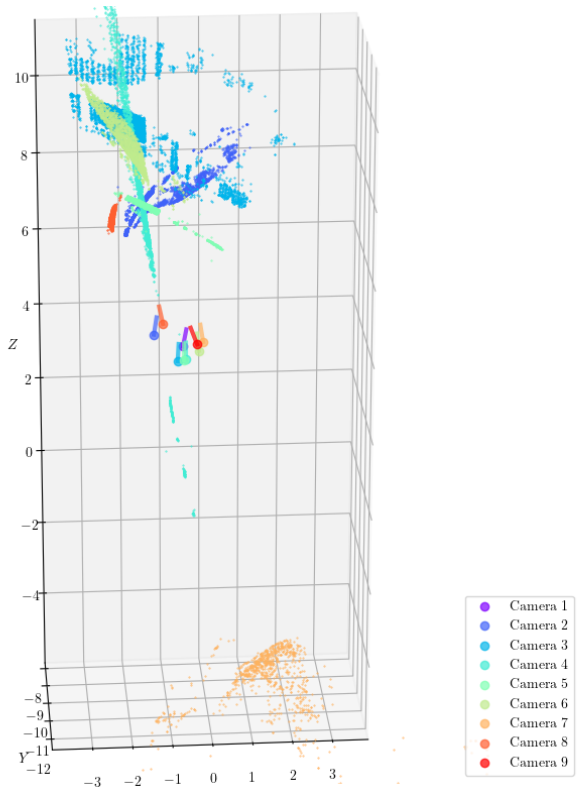

Figure 13: Reconstruction of data set 9 before LM.

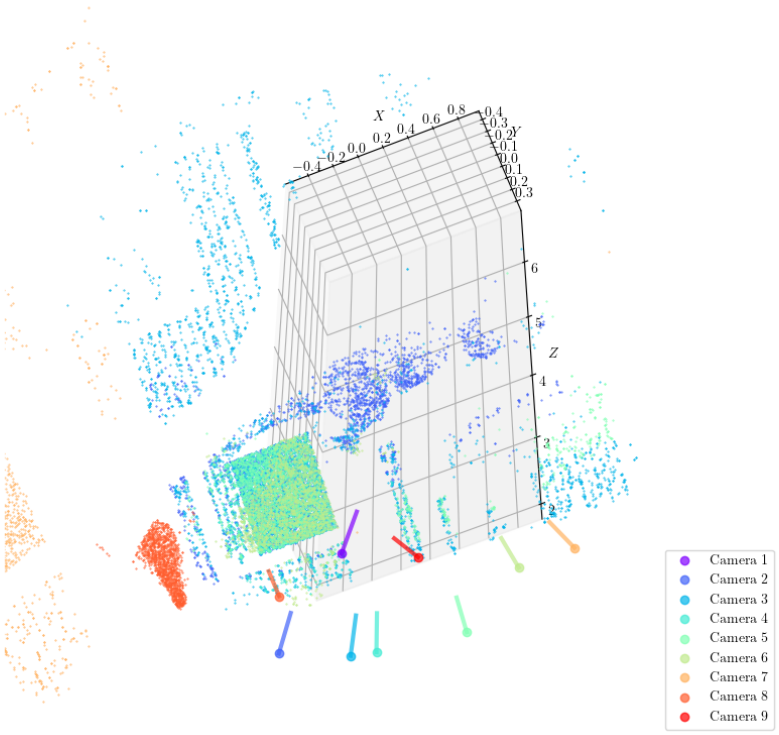

Figure 14: Reconstruction of data set 9 after LM. 4 point clouds align out 7 possible.

Reconstruction of the data sets with `T_robust=False`

Figures 15-28 show reconstructions of the data sets before and after LM optimization for non-robust translations.

In data sets 3 and 4, the pixel-threshold for estimation of the essential matrix was the threshold of recommendation, and the threshold for estimation of the homography was 3 times the recommended threshold. For data sets 5-9 the recommended threshold was increased three times when estimating the initial essential matrix.

Data set 7 (Figure 24) was the most difficult of all data sets for non-robust translations to reconstruct. This is surprising considering that this data set was the least difficult to reconstruct for robust translations. As one can see in the figure, the different reconstructions of the fasade align are very poorly.

The reconstructions does not look great here either before optimization, and afterwards we can see that the point clouds align with each other with slight offset showing that LM has significantly improved the solutions. From inspection we can clearly see that the reconstructions here are much better than when estimating the translations robustly. The final 3D-reconstructions of the data sets are not perfect, but considering that a full-scale bundle adjustment was not implemented, highest quality of the reconstructions is not expected.

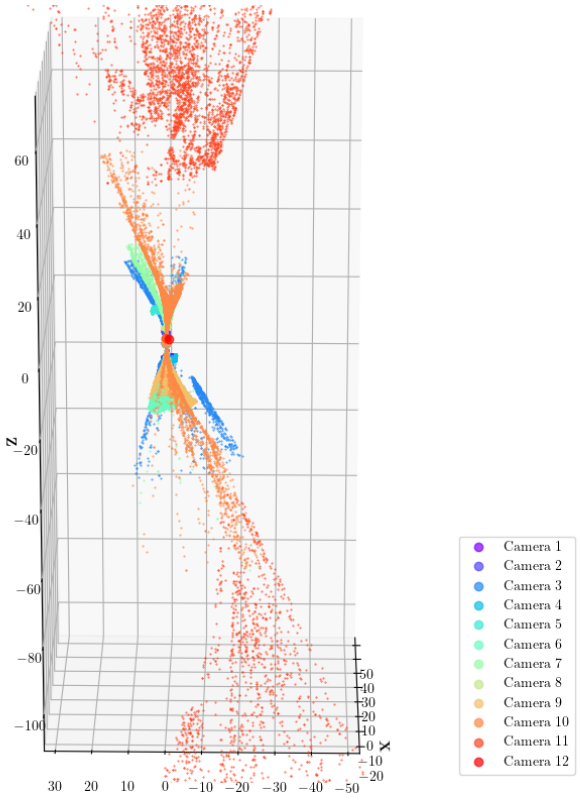

Figure 15: Reconstruction of data set 3 before LM.

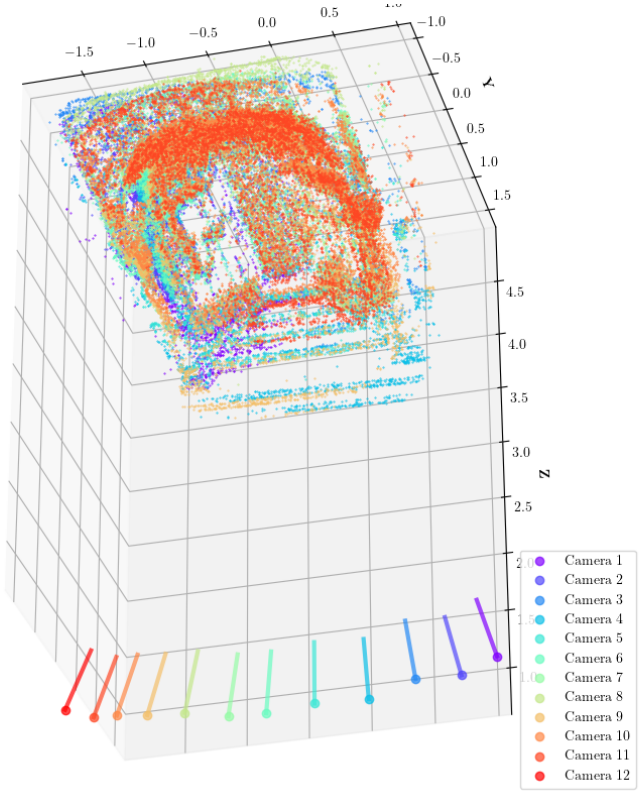

Figure 16: Reconstruction of data set 3 after LM. All point clouds align.

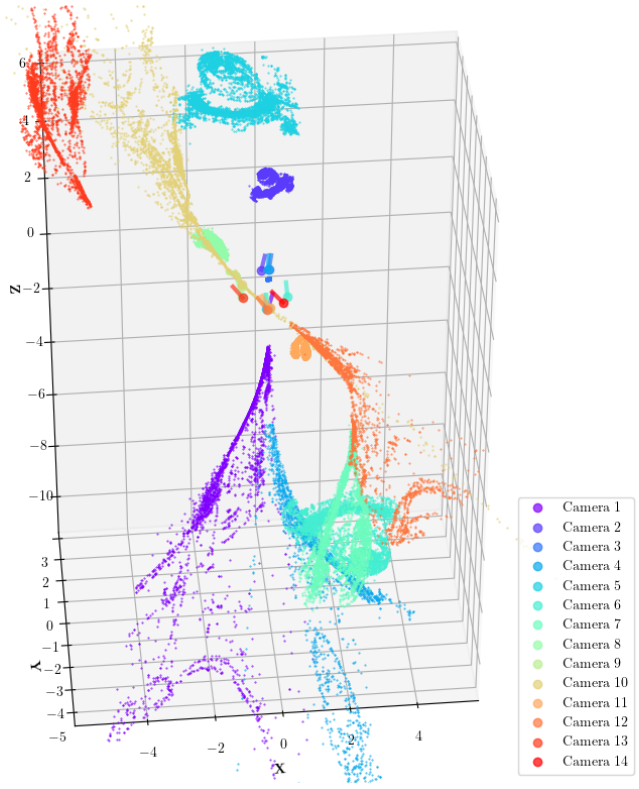

Figure 17: Reconstruction of data set 4 before LM.

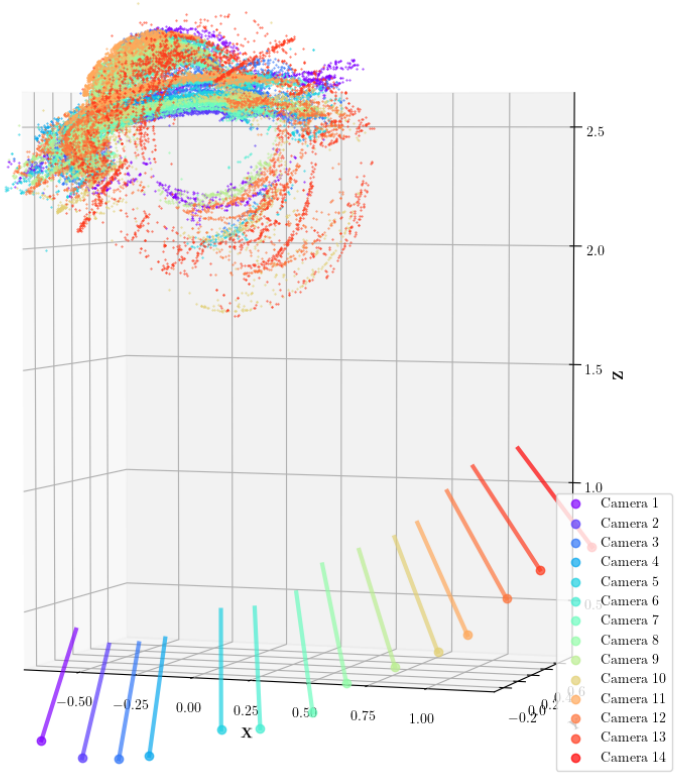

Figure 18: Reconstruction of data set 4 after LM. All point clouds align.

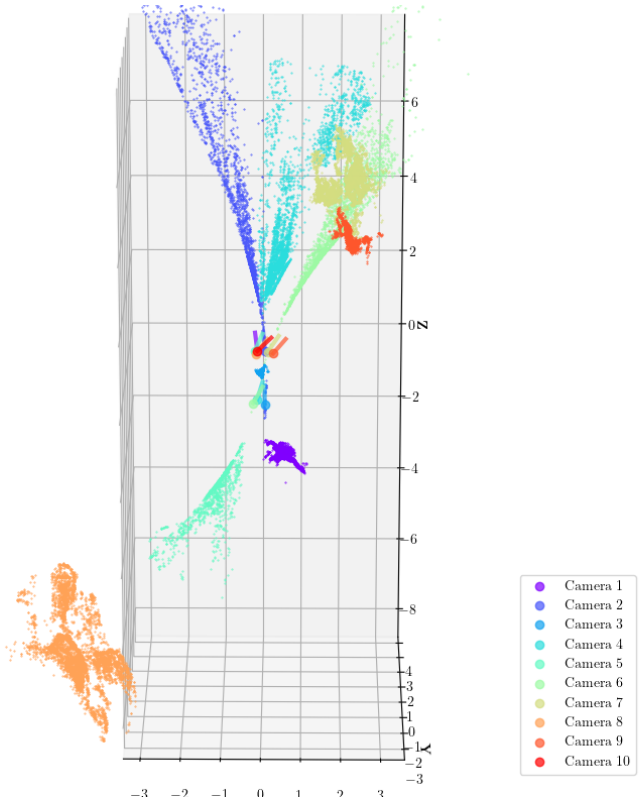

Figure 19: Reconstruction of data set 5 before LM.

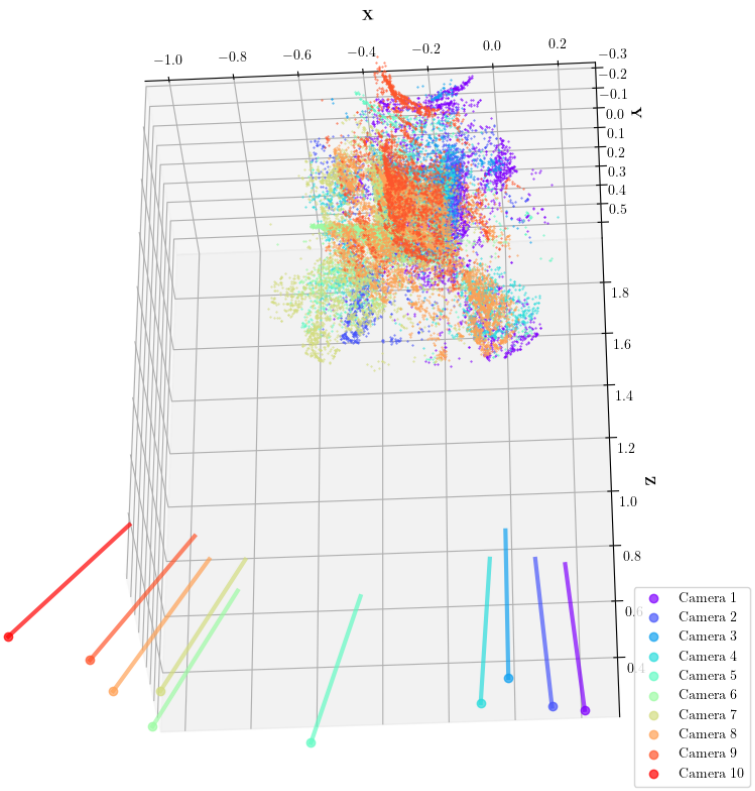

Figure 20: Reconstruction of data set 5 after LM. All point clouds align.

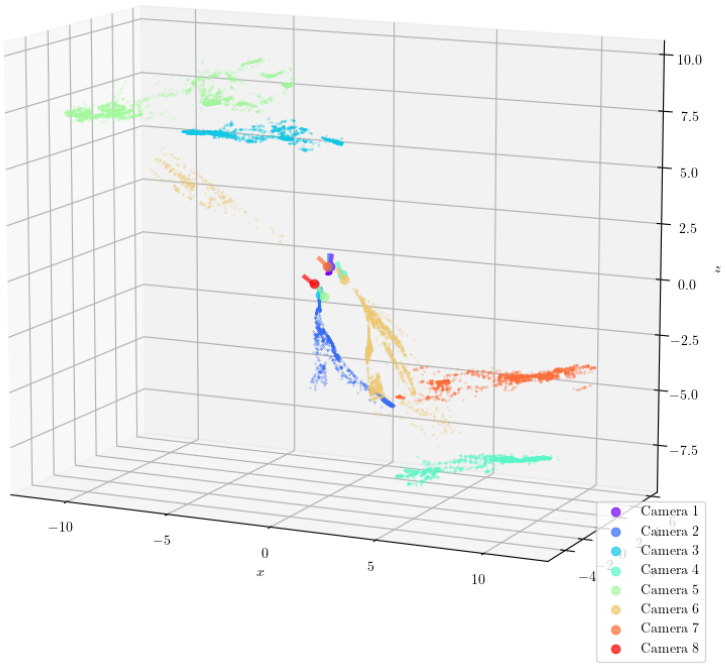

Figure 21: Reconstruction of data set 6 before LM.

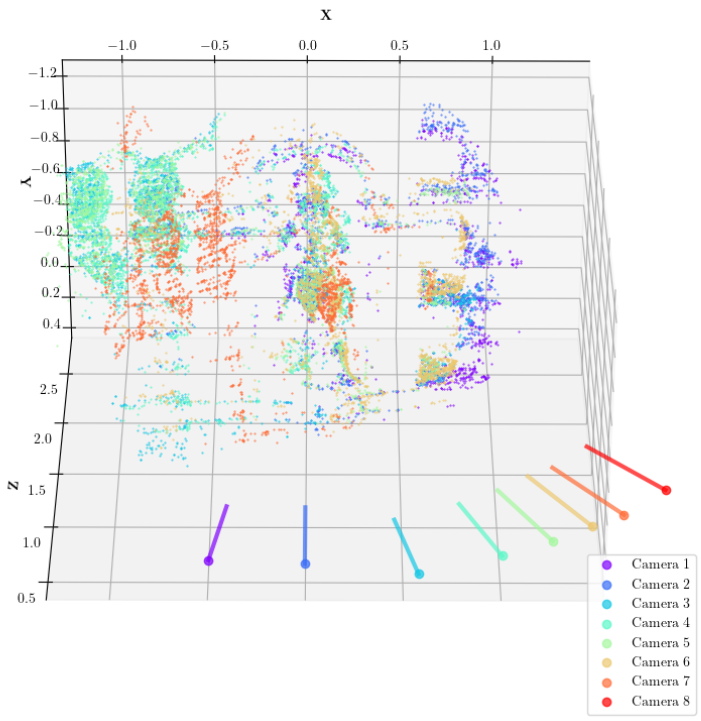

Figure 22: Reconstruction of data set 6 after LM. All point clouds align.

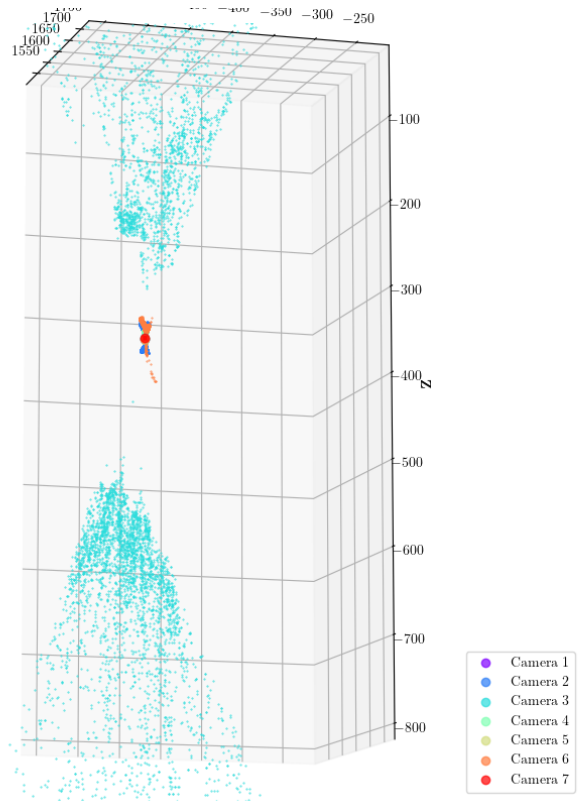

Figure 23: Reconstruction of data set 7 before LM.

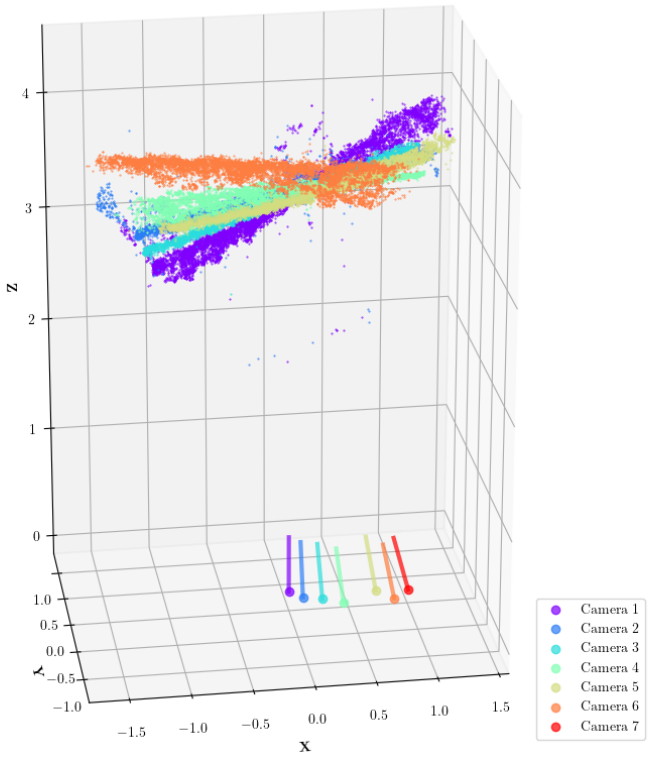

Figure 24: Reconstruction of data set 7 after LM, view from above. The most difficult data set for non-robust translations to reconstruct properly.

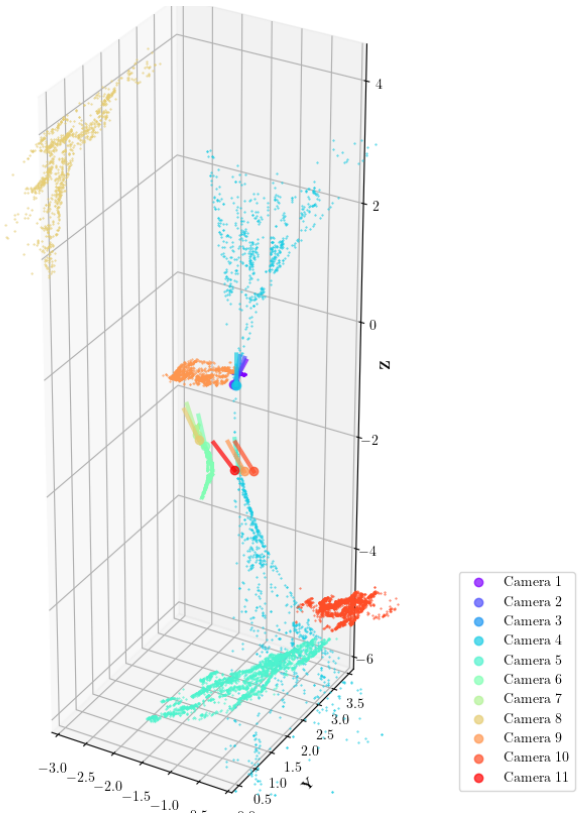

Figure 25: Reconstruction of data set 8 before LM.

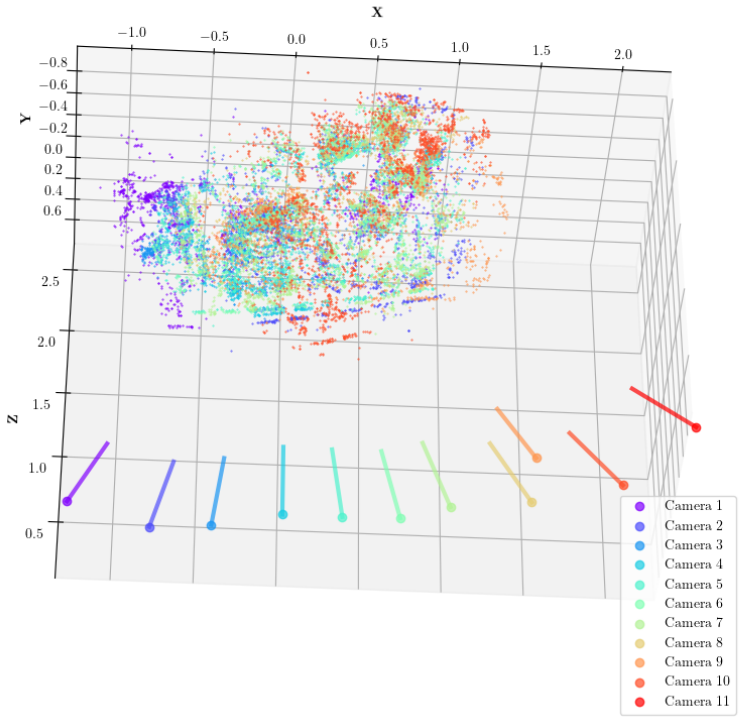

Figure 26: Reconstruction of data set 8 after LM. All point clouds align.

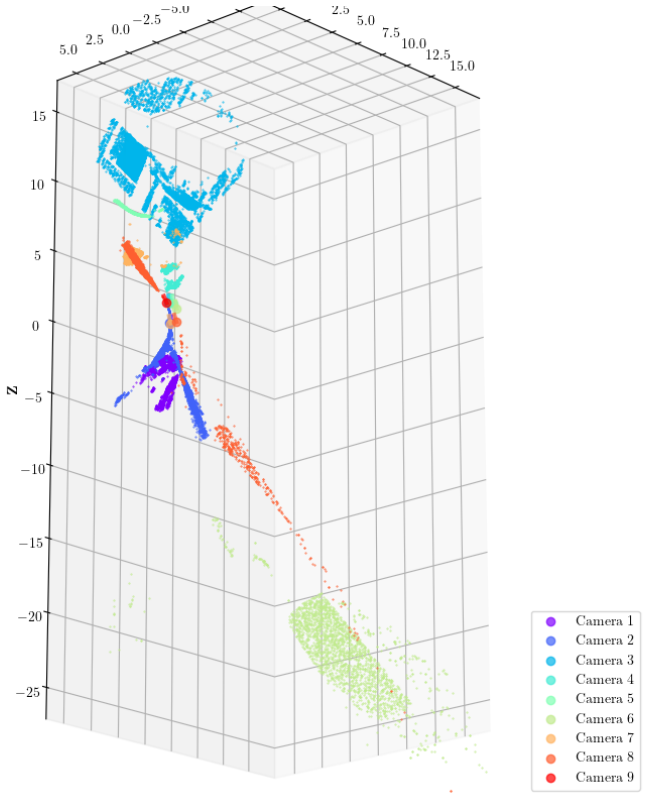

Figure 27: Reconstruction of data set 9 before LM.

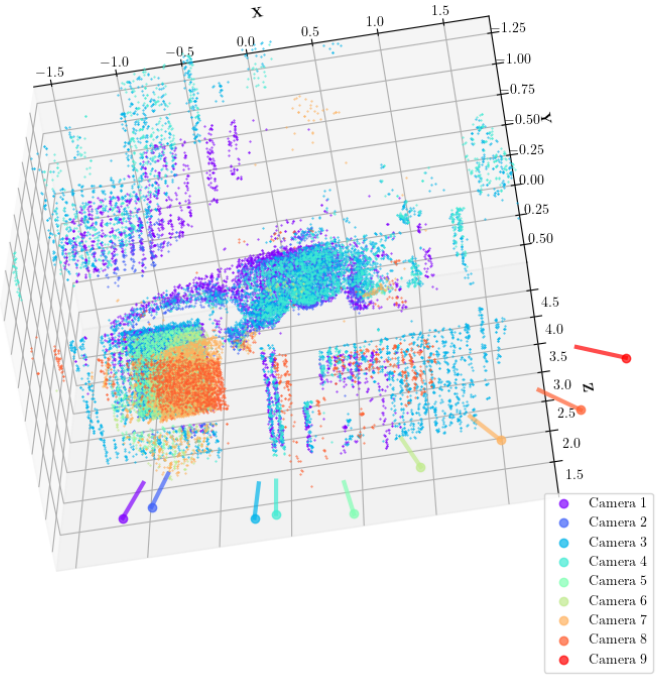

Figure 28: Reconstruction of data set 9 after LM. All point clouds align.

Conclusion

In general the implementation works, but for some parts not exactly as expected according to the instructions of the assignment. Great efforts went into trying to get `estimate_T_robust` and the modifed LM from assignment 4 to work properly for all data sets but without success. Some compromises were made along the way as last resorts, and at the end the data sets were beautifully reconstructed.