Buildings built in minutes - SfM
RBE549 Project 2

Sumukh Porwal, Piyush Thapar
MS Robotics Engineering
Worcester Polytechnic Institute
Email: sporwal@wpi.edu, pthapar@wpi.edu

***Abstract*—This report presents the results of implementing a Structure from Motion (SfM) pipeline. The SfM pipeline estimates the camera poses and 3D structure of a scene from a set of 2D images by creating point clouds, which recreate the scene in 3D.**

I. Phase 1: Structure from Motion

A. Introduction

Structure from motion is a computer vision technique that aims to recover the 3D structure of a scene from a set of 2D images. The basic idea is to estimate the camera poses and 3D points that best explain the observed 2D points in the images. This is a challenging problem, as it requires solving for the camera poses and 3D points simultaneously, and is sensitive to noise and outliers. In this phase, we implemented a basic structure from motion pipeline that estimates the camera poses and 3D points from a set of 2D images. We begin this section with the math and algorithms used in a basic SfM pipeline for a single pair of images, and then extend it to multiple images. We then discuss the challenges and limitations of the basic pipeline, and propose a more robust and scalable pipeline that addresses these issues. Finally, the results of the pipeline applied to Unity Hall at WPI are presented.

B. Dataset

We are given with a set of 5 images of Unity Hall at WPI as shown in fig. 1, using a Samsung S22 Ultra's primary camera at f/1.8 aperture, ISO 50 and 1/500 sec shutter speed. The camera is calibrated after resizing using a Radial-Tangential model with 2 radial parameters and one tangential parameter using the MATLAB R2022a's Camera Calibrator Application beforehand. The images provided are already distortioncorrected and resized to 800 × 600px.

Fig. 1. Dataset

C. Estimating the Fundamental Matrix

The first step in the Structure from Motion (SfM) pipeline is to estimate the fundamental matrix. The fundamental matrix describes the epipolar geometry between two images, relating the points in one image to the corresponding epipolar lines in the other image. This relationship is captured in the following mathematicaly expression, refered to as the epipolar constraint:

$$\mathbf{x}_i'^{\mathbf{T}}\mathbf{F}\mathbf{x}_i = 0$$

where $\mathbf{x}_i$ and $\mathbf{x}_i'$ are the homogeneous coordinates of the corresponding points in the two images, and $\mathbf{F}$ is the fundamental matrix. This equation is then expanded to the following form:

$$\begin{bmatrix} x_i' & y_i' & 1 \end{bmatrix} \begin{bmatrix} f_{11} & f_{12} & f_{13} \\ f_{21} & f_{22} & f_{23} \\ f_{31} & f_{32} & f_{33} \end{bmatrix} \begin{bmatrix} x_i \\ y_i \\ 1 \end{bmatrix} = 0$$

which can be transformed into a the following linear equation:

$$\begin{bmatrix} x_1x_1' & x_1y_1' & x_1 & y_1x_1' & y_1y_1' & y_1 & x_1' & y_1' & 1 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ x_mx_m' & x_my_m' & x_m & y_mx_m' & y_my_m' & y_m & x_m' & y_m' & 1 \end{bmatrix} \begin{bmatrix} f_{11} \\ f_{21} \\ f_{31} \\ f_{12} \\ f_{22} \\ f_{32} \\ f_{13} \\ f_{23} \\ f_{33} \end{bmatrix} = 0$$

We need at least 8 point correspondences to solve for equation above, as each correspondence only contributes 1 contraint to the system, as the epipolar constraint is a scalar equation.

To solve the equation we use singular value decomposition (SVD) to find the least squares solution to the linear equation. The fundamental matrix is then constructed from the least squares solution, and the rank-2 constraint is enforced by setting the smallest singular value to zero.

To increase the stability of the solution, the 8 points from each image are normalized using basic normalization matrices, $\mathbf{T}$ and $\mathbf{T}'$. Once the eight-point algorithm is applied to the normalized points, the fundamental matrix is denormalized using the following equation:

$$\mathbf{F}' = \mathbf{T}'^{\mathbf{T}}\mathbf{F}\mathbf{T}$$

It is important to note, however, that the eight-point algorithm is sensitive to noise and requires a sufficient number of point correspondences for accurate results. Additionally, it can be affected by degenerate configurations, such as coplanar points or degenerate camera motion. To solve these solutions one more step is required. The resulting fundamental matrix can be visualized by looking at the epipolar lines in the two images. Epipolar lines are lines which pass though a feature point one image and location of the other camera in that image. The epipolar lines are shown in Figure 2.

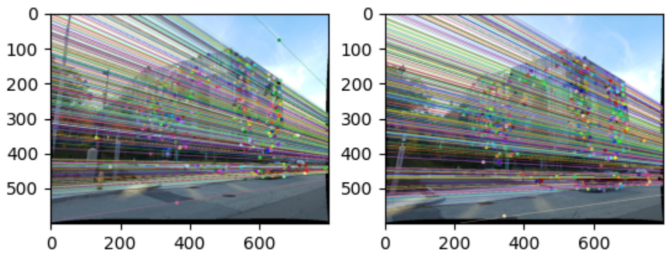

Fig. 2. Epipolar lines in first 2 images

D. Match Outlier Rejection via RANSAC

To address the aforementioned issues with the naive 8-point algorithm, we use the RANdom SAmple Consensus (RANSAC) algorithm to reject outlier correspondences, and thus obtain a more accurate estimate of the fundamental matrix. RANSAC is an iterative algorithm that selects a random subset of the data and fits a model to that subset, in our case a random set of 8 correspondences. It then evaluates the model on the remaining data, and the points that are consistent with the model are considered inliers. This process is repeated for a specified number of iterations, and the model with the most inliers is chosen as the best estimate.

One aspect of the algorithm which was glossed over is the evaluation of the model on the remaining data. The approach is given in pseudo code below,

```
n=0;
for $i$ = $1{:}M$ do
    // Choose 8 correspondences, $\hat{x}_1$ and $\hat{x}_2$ randomly
    F = EstimateFundmentalMatrix($\hat{x}_1$, $\hat{x}_2$);
    $\mathcal{S} = \emptyset$;
    for $j$ = $1{:}N$ do
        if $|x_{1j}^T F x_{2j}| < \epsilon$ then
            $\mathcal{S} = \mathcal{S} \cup \{j\}$
        end
    end
    if $n < |\mathcal{S}|$ then
        $n = |\mathcal{S}|$;
        $\mathcal{S}_{in} = \mathcal{S}$
    end
end
```

E. Estimate Essential Matrix from Fundamental Matrix

The next step of the SfM pipeline is to estimate the essential matrix from the fundamental matrix. The essential matrix is a 3×3 matrix that relates the corresponding points in two images, assuming that the cameras obey the pinhole model. It can be computed from the fundamental matrix using the following relationship:

$$\mathbf{E} = \mathbf{K^T F K}$$

Due to noise in the calculation of the fundamental matrix, the essential matrix is not guaranteed to be of rank 2. To enforce this constraint, we use singular value decomposition to decompose the essential matrix into its constituent parts, and then reconstruct it using the following equation:

$$\mathbf{E} = U \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} V^T$$

F. Estimate Camera Pose from Essential Matrix

Once the essential matrix has been estimated, the next step is to estimate the camera pose. The essential matrix is a representation of the relative pose between the two cameras, and it can be decomposed into the rotation and translation components. The decomposition of the essential matrix is given by the following equation:

$$\mathbf{E} = [\mathbf{t}]_\times \mathbf{R}$$

The decomposition of the essential matrix is not unique, and will result in four possible camera poses given below,

1) $C_1 = U(:,3)$ and $R_1 = UWV^T$
2) $C_2 = -U(:,3)$ and $R_2 = UWV^T$
3) $C_3 = U(:,3)$ and $R_3 = UW^TV^T$
4) $C_4 = -U(:,3)$ and $R_4 = UW^TV^T$

Also if $det(R) = -1$, then the camera pose must be corrected i.e. $C = -C$ and $R = -R$.

G. Triangulation Check for Cheirality Condition

To resolve this ambiguity and find a single camera pose, we use the following method:

1) Linear Triangulation: With two camera poses (R_i, C_i) and the corresponding 2D points in the images (x_i, x_i') we can calulate X, or the world point of each correspondence. To find a solution to the problem of triangulation, we begin with the pinhole projection model:

$$\begin{bmatrix} \mathbf{x} \\ 1 \end{bmatrix} = \alpha \mathbf{P} \begin{bmatrix} \mathbf{X} \\ 1 \end{bmatrix}$$

where $\mathbf{P}$ is the projection matrix:

$$\mathbf{P} = \mathbf{K}[\mathbf{R}|\mathbf{t}]$$

which is equivalent to

$$\left[\begin{bmatrix} \mathbf{x} \\ 1 \end{bmatrix}_\times \mathbf{P} \right] \begin{bmatrix} \mathbf{X} \\ 1 \end{bmatrix} = 0$$

We stack the above equation for each camera pose and its corresponding image point, and then solve for X using singular value decomposition. The result of this process is shown in Figure 3.

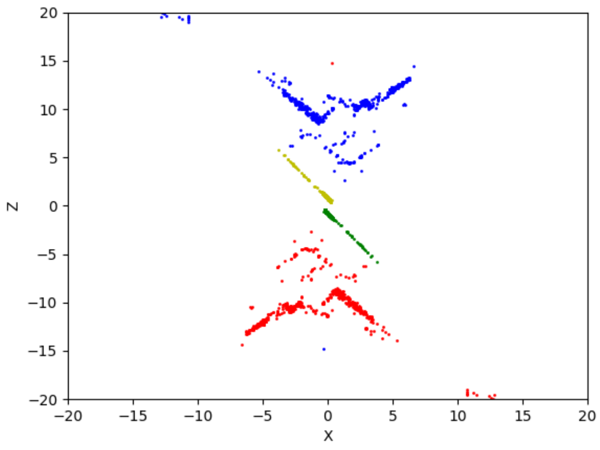

Fig. 3. Possible World Points using first 2 camera pose

2) Enforce the cheirality condition: The cheirality condition is simply the condition that the 3D points are in front of the camera. This is enforced by checking the sign of the depth of the 3D points, and discarding the camera poses that do not satisfy the condition. The cheirality condition is given by the following equation:

$$\mathbf{R}(:,3)^T(X - \mathbf{C}) > 0$$

After enforcing the cheirality condition the selected camera pose is shown in Figure 4.

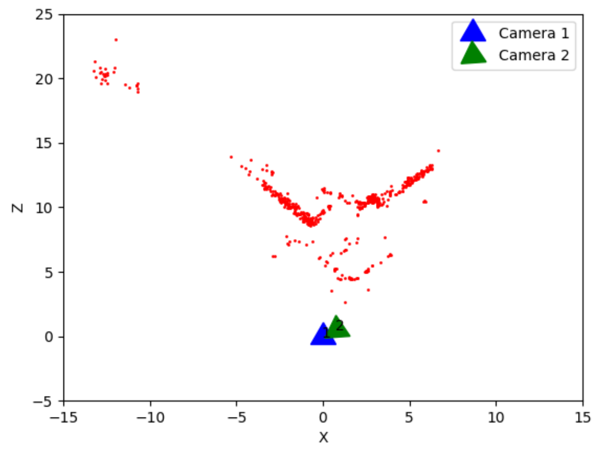

Fig. 4. Linear Triangulation

H. Non-Linear Triangulation

Given the linearly estimated 3D world points from the previous step, we refined their locations to minimize the reprojection error. The linear triangulation method minimizes the algebraic error, but the reprojection error is a more geometrically meaningful error that can be computed by measuring the geometrix error between an image points and the world point projected into its image plane. Since there are likely nonlinearities in the camera model, this is a more accurate method for estimating the 3D points. A comparison of the projections between the non-linear and linear triangulation methods is shown in Figure 5 and 6. The optimization function is given below,

$$min \sum_{j=1,2} \left(u^j - \frac{P_1^{jT} \widetilde{X}}{P_3^{jT} X} \right)^2 + \left(v^j - \frac{P_2^{jT} \widetilde{X}}{P_3^{jT} X} \right)^2$$

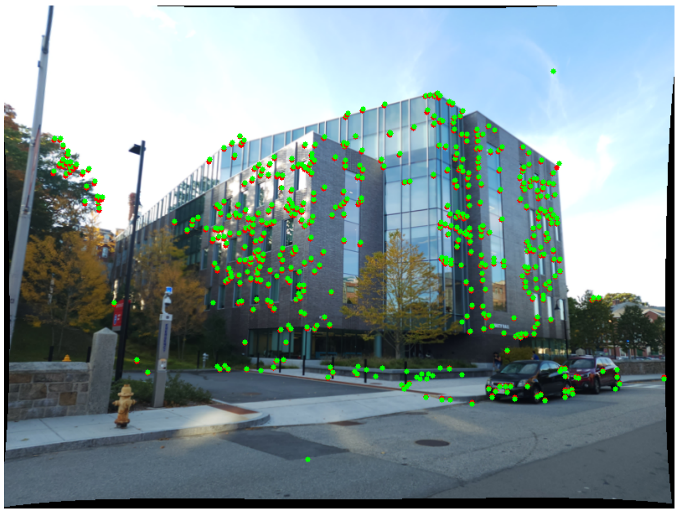

Fig. 5. Linear Reprojection

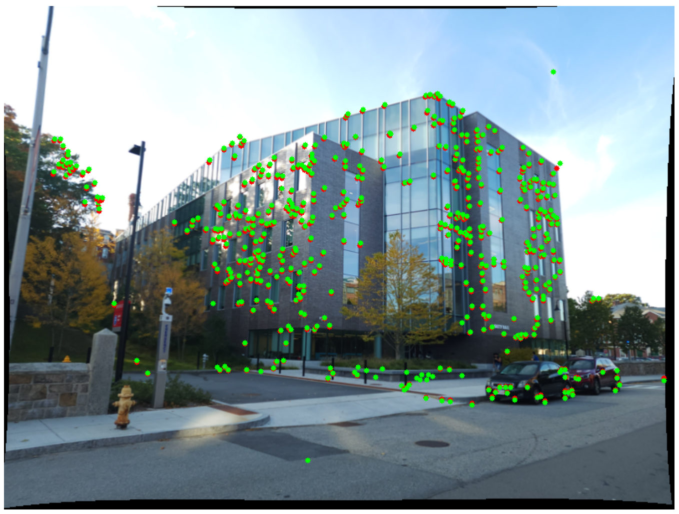

Fig. 6. Nonlinear Reprojection

Also the Linear and NonLinear Triangulation estimates are shown in the Figure 7.

I. Perspective-n-Points

1) Linear Camera Pose Estimation: To extend this pipeline to multiple images, we need to estimate the camera pose for

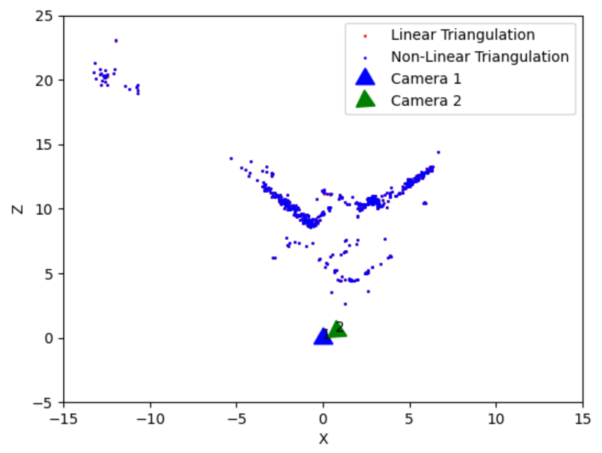

Fig. 7. Linear and NonLinear Triangulation

each image. This is accomplished using the Perspective-n-Points (PnP) algorithm, which estimates a camera pose from a set of 3D points and their corresponding 2D projections. The PnP algorithm can take in n, hence the name, but we used n = 6 which allows the math to remain simple. PnP first consists of forming a linear estimate of the camera pose using, formed by transforming

$$\begin{bmatrix} \mathbf{x} \\ 1 \end{bmatrix} = \alpha \mathbf{P} \begin{bmatrix} \mathbf{X} \\ 1 \end{bmatrix}$$

into the following form:

$$\begin{bmatrix} X, Y, Z, 1, 0, 0, 0, 0, -xX, -xY, -xZ, -x \\ 0, 0, 0, 0, X, Y, Z, 1, -yX, -yY, -yZ, -y \end{bmatrix} \begin{bmatrix} p_{11} \\ p_{12} \\ p_{13} \\ p_{14} \\ p_{21} \\ p_{22} \\ p_{23} \\ p_{24} \\ p_{31} \\ p_{32} \\ p_{33} \\ p_{34} \end{bmatrix} = \mathbf{0}$$

and then stacking it n times for each point used in the linear PnP. This is then solved using SVD, yielding the projection matrix $\mathbf{P}$. $\mathbf{P}$ is then decomposed into $\mathbf{R}$, and $\mathbf{t}$:

$$\mathbf{R} = \mathbf{K}^{-1}\mathbf{P}(:, 1:3)$$

$$\mathbf{t} = \mathbf{K}^{-1}\mathbf{P}(:, 4)$$

which is then simplified:

$$\mathbf{U}, \mathbf{D}, \mathbf{V} = \mathrm{SVD}(\mathbf{R})$$

$$\mathbf{R} = \mathbf{U}\mathbf{V}^{\top}$$

$$\mathbf{t} = \mathbf{t}/\mathbf{D}_{11}$$

Finally, if $det(R) = -1$, we flip the sign of of $\mathbf{R}$ and $\mathbf{t}$.

2) PnP RANSAC: This process described above yields a decent estimate of the camera pose, but is very sensitive to noise and outliers. To address this the RANSAC algorithm is applied, with 6 random points chosen each iteration, and with inliers being determined using reprojection error. The pseudocode is shown below

```
$n = 0$
for $i$ = $1{:}M$ do
    // Choose 6 correspondences, $\hat{X}$ and $\hat{x}$, randomly
    [C R] = LinearPnP($\hat{X}$, $\hat{x}$, K);
    $\mathcal{S} = \emptyset$;
    for $j$ = $1{:}N$ do
        // Measure Reprojection error
        $e = \left(u - \frac{P_1^T \tilde{X}}{P_3^T \tilde{X}}\right)^2 + \left(v - \frac{P_2^T \tilde{X}}{P_3^T \tilde{X}}\right)^2$;
        if $e < \epsilon_r$ then
            $\mathcal{S} = \mathcal{S} \cup \{j\}$
        end
    end
    if $n < |\mathcal{S}|$ then
        $n = |\mathcal{S}|$;
        $\mathcal{S}_{in} = \mathcal{S}$
    end
end
```

J. Nonlinear PnP

While the linear PnP RANSAC algorithm is a good start, it is not perfect. To address this, we use the LevenbergMarquardt algorithm to refine the camera pose estimate. The Levenberg-Marquardt algorithm is a non-linear optimization algorithm that minimized a sum of squares of residuals. The residuals for the algorithm are simply given by the difference between the reprojected points and the true image points:

$$\mathbf{r} = \mathbf{x} - \mathbf{P}\mathbf{X}$$

which means that Levenberg-Marquardt is minizing the following cost function:

$$\min_{C,q} \sum_{i=1,J} \left(u^j - \frac{P_1^{jT}\widetilde{X}_j}{P_3^{jT}\widetilde{X}_j}\right)^2 + \left(v^j - \frac{P_2^{jT}\widetilde{X}_j}{P_3^{jT}X_j}\right)^2$$

where $\widetilde{X}$ is the homogeneous representation of X. P_i^T is each row of camera projection matrix, P which is computed by $P = KR[I_{3\times3} - C]$. A compact representation of the rotation matrix using quaternion is a better choice to enforce orthogonality of the rotation matrix, $R = R(q)$ where q is four dimensional quaternion.,

The result of this process is shown in Figure 8.
Comparison of reprojected points between non-linear and linear PnP for Camera 4.

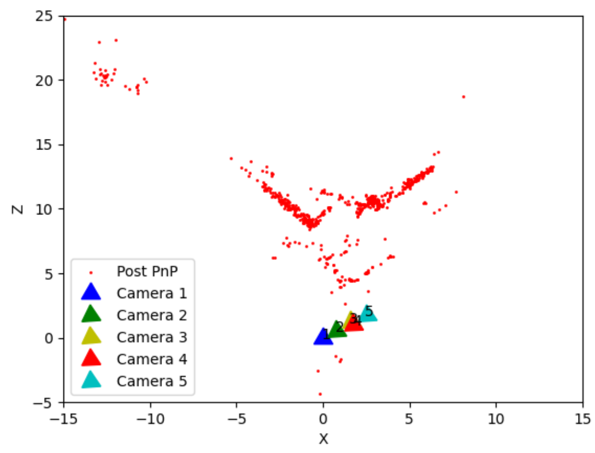

Fig. 8. Output from 5 different view of Unity Hall, without bundle adjustment

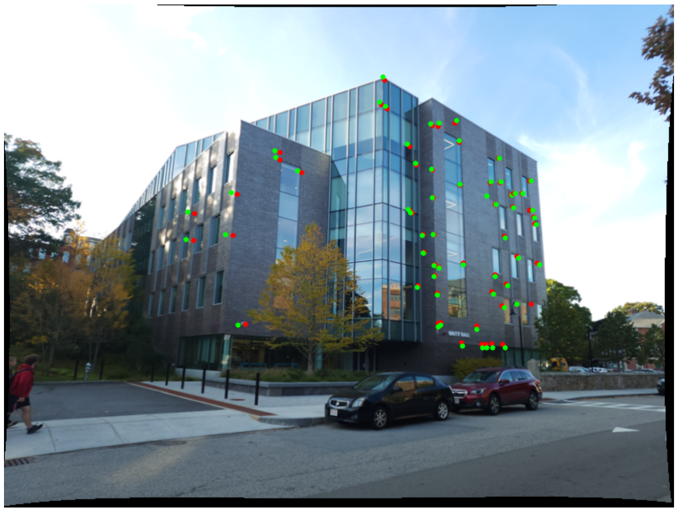

Fig. 9. Linear PnP Reprojection for Camera 4

	Error
Linear reprojection	6.8634
Nonlinear reprojection	6.8303
Linear PnP (1,3)	803.03
Nonlinear PnP (1,3)	726.12
Linear PnP (1,4)	19.27
Nonlinear PnP (1,4)	6.89
Linear PnP (1,5)	33.879
Nonlinear PnP (1,5)	4.501

The reprojection errors are listed in the above table

K. Visibility Matrix and Bundle Adjustment

1) Visibility Matrix: We created a visibility matrix for the given number of cameras and total world points. The function initializes the matrix with zeros and iterates over each world point and camera to mark the visibility of the world point in each camera.

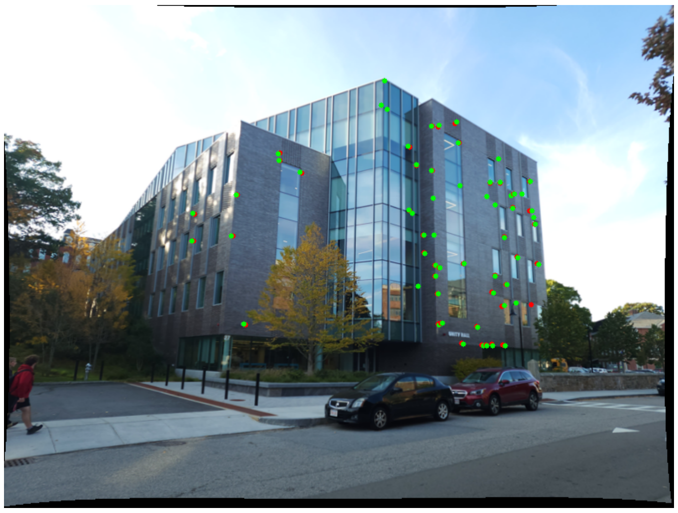

Fig. 10. Nonlinear PnP Reprojection for Camera 4

2) Bundle Adjustment: Now, to reduce the projection error simultaneously for all the 5 images using this visibility matrix. Bundle adjustment takes input parameters related to camera poses, 3D points, visibility, and intrinsic matrix and uses least squares optimization to refine the camera poses and 3D points.

$$\min_{\{C_i,q_i\}_{i=1}^{i},\{X\}_{j=1}^{J}} \sum_{i=1}^{I}\sum_{j=1}^{J} V_{ij}\left(\left(u^j - \frac{P_1^{jT}\tilde{X}}{P_3^{jT}\tilde{X}}\right)^2 + \left(v^j - \frac{P_2^{jT}\tilde{X}}{P_3^{jT}\tilde{X}}\right)^2\right)$$

The function returns the optimized camera poses and 3D points. This is shown in Figure 11, which shows bundle adjustment for all the camera views.

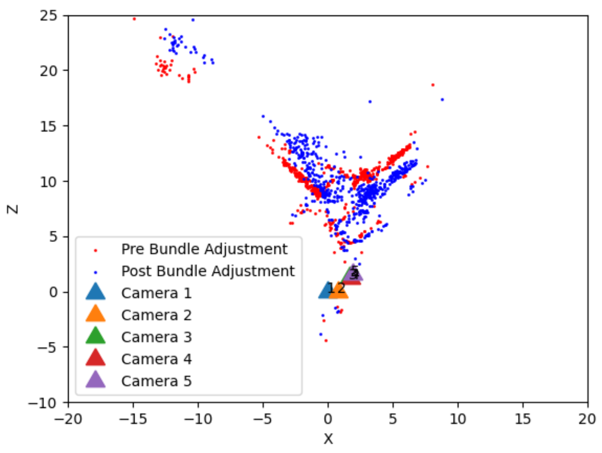

Fig. 11. Camera poses and World Coordinates after Bundle Adjustment