CS549: Project 2 - SFM and NeRf using 1 late day

Siyuan Huang
shuang4@wpi.edu

Yijia Wu
ywu21@wpi.edu

I. Phase 1: Monocular Camera Structure from Motion

In this section, a detailed analysis of our implementation of the monocular camera structure from motion(SFM) will be presented. Our implementation of the SFM pipeline consists of 7 major steps: 1) Feature matching with SIFT. 2) Estimating fundamental matrix with epipolar constraints. 3) Estimating essential matrix from fundamental matrix. 4) Estimating camera pose from essential matrix. 5) Linear triangulation and nonlinear triangulation with Cheriality constraints checking. 6) Solve for perspective-n-points using linear and nonlinear optimization. 7) Bundle adjustment for all input images.

A. *Feature Matching with SIFT*

The first step in our pipeline is to obtain feature matches between each pair of monocular camera images. We used SIFT as our feature-matching algorithm. Prior to finding the feature matches between each pair of input images, the camera intrinsic matrix has been found through a calibration procedure and then the raw images have been rectified to account for distortion. Figure 1 shows the feature matches found between the first and the second image.

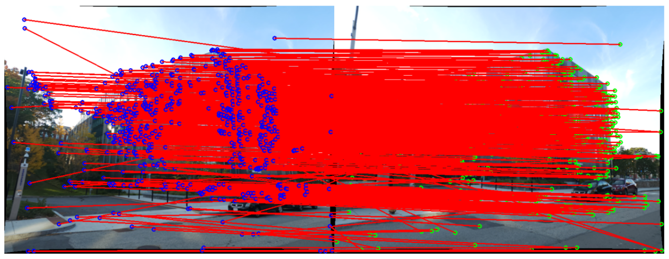
Fig. 1: Feature Matches found through SIFT

B. *Estimating Fundamental Matrix with Epipolar constraints*

The second step of our pipeline was to estimate the fundamental matrix between a pair of images using Epipolar constraint. Through experiments, we have found that using two layers of match outliers rejection, once with homography RANSAC and then with fundamental matrix(8-point algorithm with SVD clean-up that enforces rank=2 constraint) RANSAC, provided not only faster but also more consistent and accurate results. The threshold condition for fundamental matrix RANSAC is $|x_{2j}^T F x_{1j}| < \epsilon$. Where x_{2j} is the image coordinates of a feature in the second image, x_{1j} is the image coordinates of the same feature in the first image. F is the fundamental matrix between the two images calculated by 8-point algorithm. We believe this is because homography RANSAC first reduced the search space of the fundamental matrix RANSAC to a local minima that contains the global minima. Figure 2 shows the feature matches of a pair of images that first filtered by homography RANSAC. Figure 3 shows the final result of the feature match inliers after RANSAC with fundamental matrix. Once the inliers have been found, a final estimated fundamental matrix was calculated using the same 8-point algorithm but with all the inliers.

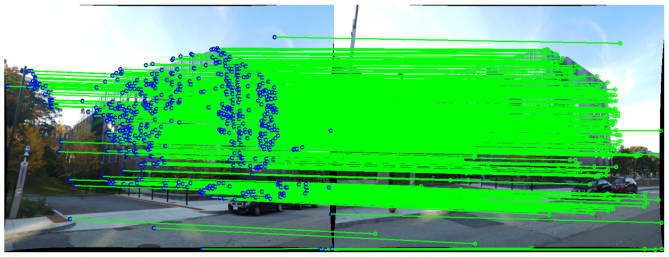
Fig. 2: Feature match inliers after homography RANSAC

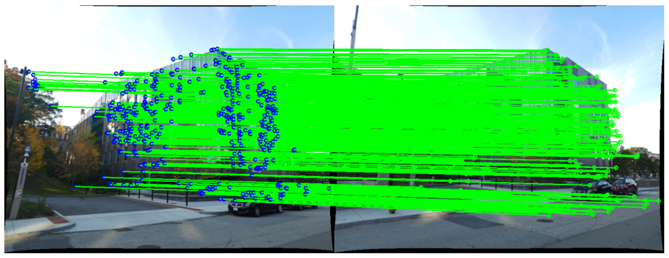
Fig. 3: Feature match inliers after 8-point algorithm RANSAC

Estimating Essential Matrix from Fundamental Matrix. Using the estimated fundamental matrix found in the previous step, we then calculated the Essential Matrix between an image pair. Since we know the camera intrinsic K from camera calibration, we can calculate the Essential Matrix $E = K^T F K$. However, since there might still be noises in the calculated Essential Matrix, we have to enforce the epipolar constraint by reconstructing the Essential Matrix with singular value decomposition with rank reduction. $E = UDV^T$ then substitute D with $\begin{bmatrix}1 & 0 & 0\\0 & 1 & 0\\0 & 0 & 0\end{bmatrix}$

C. Estimating camera pose from Essential Matrix.

With the calculated Essential matrix above $E = UDV^T$, let $W = \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}$., we can calculate the four camera configurations as:

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) $C_4 = -U(:,3) and R_4 = UWV^T$

D. Linear triangulation and nonlinear triangulation with Cheriality constraints checking

With the four possible camera pose configurations found through decomposing essential matrix in the previous step. Next we used linear triangulation with Cheriality constraints to rule out the three impossible camera pose configuration. Figure 4 shows the linear triangulation result of the top-down view of the features' world coordinates in the four camera pose configurations. To disambiguate the camera poses, we then selected the pose that had the most number of features satisified Cheriality constraints. Figure 5 shows the features' wolrd coordinates with the selected camera pose. Once we obtained the actual camera pose configuration, we then used non-linear optimization to minimize the reprojection error to improve the features' world coordinates estimation. Figure 6 shows the comparison between the result of linear triangulation and non-linear triangulation. It can be seen that non-linear traingulation significantly improved the accuracy of the estimated features' world coordinates.

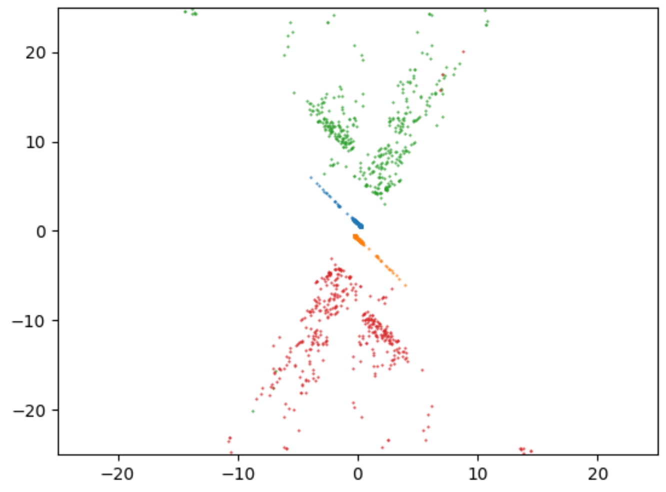

Fig. 4: Linear triangulation results for 4 possible camera pose configurations

E. Solve for perspective-n-points(PnP) using linear and non-linear optimization.

Up to now we have obtained an estimated features' world coordinates using a pair of monocular camera images. To register a third or more images' features to the constructed

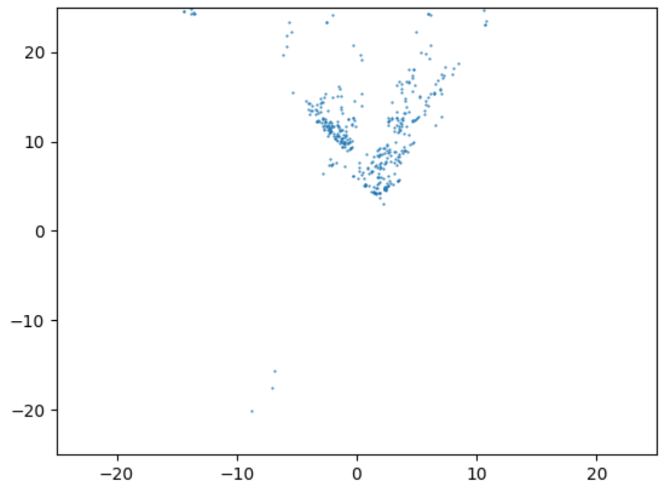

Fig. 5: Features' world coordinates in the selected camera pose

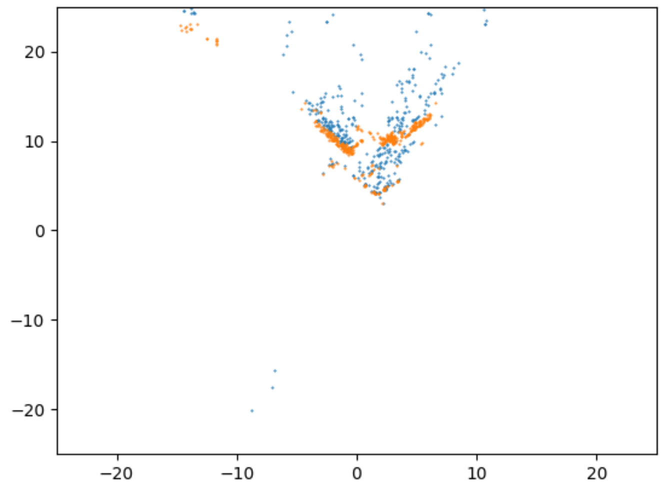

Fig. 6: Comparasion between linear and non-linear triangulation

scene, we first used PnP RANSAC to remove the outlier features in the third image, which minimize the algebraic error between the measured and reprojected features. Using the remaining inlier features, we calculated the estimated camera pose of the third image. Figure 8 shows the measured(SIFT) and reprojected features using the camera pose found with linear PnP. Due to the nature of nonlinearity of division and reprojection in our system, the estimated camera poses are inaccurate. Using the estimated camera poses from linear PnP as initial guess, we then performed non-linear optimization, which minimize the geometric reprogection error, to obtain more accurate estimation of the camera poses. Figure 8 shows the measured and reprojected features using the camera pose found with non-linear PnP. The reprojected features are much more closer to the measured featured with nonlinear PnP.

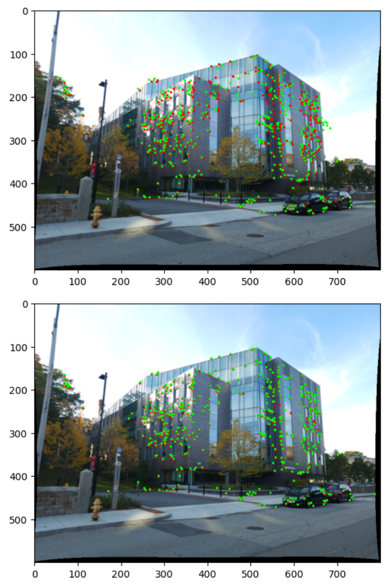
Fig. 7: Features reprojection with linear and nonlinear Triangulation

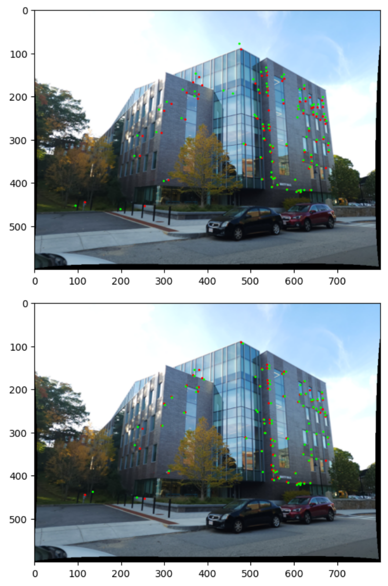
Fig. 8: Features reprojection with linear and nonlinear PnP

F. *Bundle adjustment for all input images*

Once we have registered all the images to the scene with linear and nonlinear PnP, now we need to refine the poses and 3D points together with bundle adjustment. With camera poses and 3D points calculated with previous steps as the initial guess, we used $scipy.optimize.leastsq$ to minimize the reprojection error. Figure 9 shows the result of bundle adjustment.

II. Phase 2: Neural Radiance Fields (NeRF)

For the deep learning part of this project, we implemented a simplified version of NeRF [1]. The main simplifications and

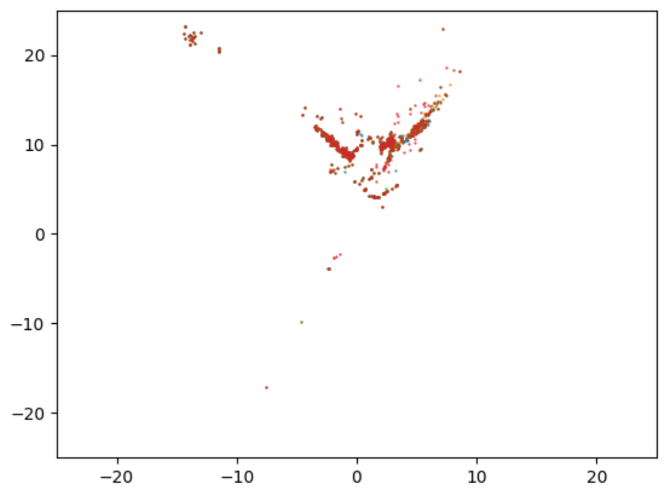
Fig. 9: Bundle Adjustment Result

TABLE I: Reprojection error (pixel)

Linear Triangulation	54.9584
Nonlinear Triangulation	5.3380
Linear PNP	154.4969
Nonlinear PNP	19.9069

modifications are listed as follows which are also corresponding to some challenges that we have encountered.

- We only used one network with dense sampling in each ray instead of using two networks.
- The density output is based on a separate linear layer which is different from the linear layer to generate the internal feature output.
- For the first 500 iteration, we only randomly choose the rays in the center area of the images.

The first challenge is that when we used a small sampling rate with a single network, even after thousands of iterations, the render result is still almost fully white. It was solved by increasing the sampling rate to 1000. If we change to use two networks, the sampling rate will not be that large.

The second challenge is that when we made the density and internal feature output shares the same linear layer and added some redundant ReLu layers, part of the render output is gray. It was solved by the second modification.

The last modification was added to reduce the training time. As a result, after 10000 iterations with sampling 4096 rays per iteration, we trained a model that can render as shown in Fig. 10. The other hyperparameters are chosen the same as what is used in the NeRF paper[1].

References

[1] B. Mildenhall, P. P. Srinivasan, M. Tancik, J. T. Barron, R. Ramamoorthi, and R. Ng, "NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis." arXiv, 2020. doi: 10.48550/ARXIV.2003.08934.

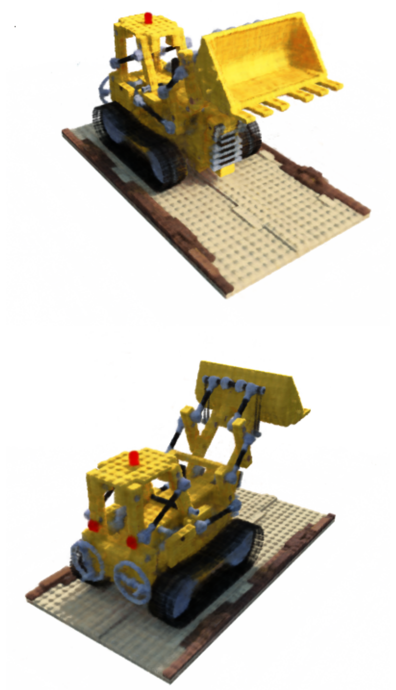

Fig. 10: Testing result for lego dataset