RBE/CS549 Computer Vision

Project 3- SfM and NeRF

Anagha Dangle - Mihir Kulkarni
ardangle@wpi.edu - mmkulkarni@wpi.edu

***Abstract*—The goal of this project is to estimate a three-dimensional structure from two-dimensional image sequences that are linked by changes in camera motion (orientation and translation). This is commonly referred to as Structure from motion. There are several algorithms that accomplish this. We hope to learn how to recreate 3D structures from a given dataset of 2D images in this project. The assignment is divided into two phases:**
Phase 1: Traditional approach- In this phase traditional approach to Structure from Motion (SfM) is implemented. The detailed process for the same is explained further in the document.
Phase 2: Deep learning approach- We implement NeRF in this section using PyTorch.

***Index Terms*—Triangulation, RANSAC, Fundamental matrix, Essential Matrix, Bundle Adjustment, NeRF.**

I. Phase 1: Traditional approach

In this section, with a given set of 5 images from a monocular camera and their feature point correspondences, we reconstructed a 3D scene while also obtaining camera poses with respect to the scene.

This approach has six main basic steps:
1. Estimate Fundamental Matrix from the given SIFT feature correspondences.
2. Estimate the Essential matrix using the fundamental matrix and Estimate camera poses.
3. Refining camera pose using Cheirality condition and Linear Triangulation.
4. Calculating the Visibility Matrix
5. Performing Bundle Adjustment.

A. Fundamental Matrix Estimation

The feature point correspondences from all of the text files were parsed and converted to an appropriate format for use in the rest of the pipeline. To deal with outliers in the data, we make use of the Random Sampling Consensus (RANSAC). We estimated the inlier-feature correspondences to use in the subsequent steps by estimating the Fundamental matrix and performing RANSAC with an epipolar constraint.

Using the normalized 8-point algorithm, we computed the Fundamental matrix only for images 1 and 2. The 8-point algorithm involves using singular value decomposition to solve a linear solver matrix generated by stacking the kroenker product between 8-point correspondences.

B. Essential Matrix and Camera Pose estimation

Given the rotation and translation, the Essential matrix E connects corresponding image points from both cameras.

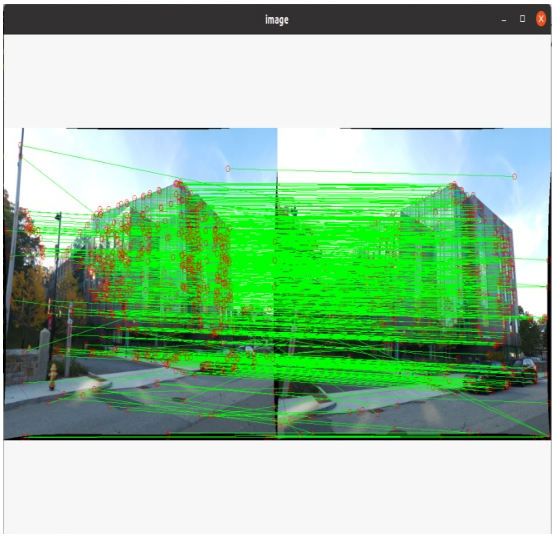

Fig. 1: RANSAC

Decomposing E yields four mathematically possible poses which are the 4 combinations of the possible camera locations with respect to the 2 image planes. The obtained Essential matrix is:- .

C. Refining Camera Pose and Linear Triangulation

To disambiguate 2 poses out of the 4 and get the 2 unique poses, we use the chierality condition. The Cheirality Condition requires that the reconstructed points be visible to the cameras. We test this condition by triangulating the 3D points using linear least squares to determine the sign of the depth Z in the camera coordinate system relative to the camera center. The camera pose that gives the maximum number of positive depth points is selected.

We can perform Linear triangulation to obtain the 3D location of a world point given a point correspondence and the projection matrix estimated using the unique disambiguate camera pose (R, C) and intrinsics (K) parameters.

D. Non-linear triangulation

Now, having obtained the 3D points from Linear triangulation from the selected pose, we can refine these 3D points by performing optimization. We try to minimize the reprojection error i.e. the geometric error is re-calculated by projecting the points on the image plane.

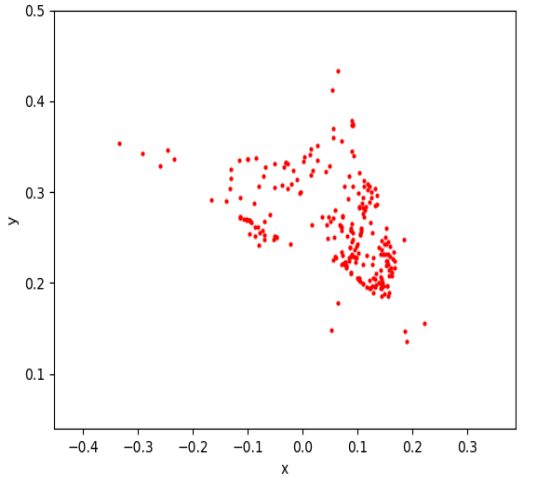

Fig. 2: Non Linear Triangulation

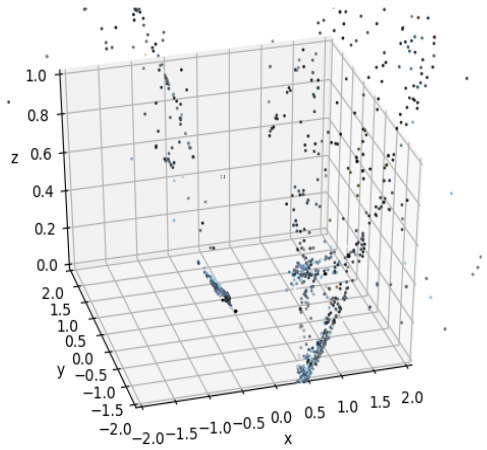

Fig. 3: Bundle Adjustment

E. Linear PnP, PnP RANSAC, and Non-Linear PnP

Given the intrinsic camera matrix K, 3D points, and their corresponding 2D points, we can compute the camera poses which is termed the perspective-n-point problem. We solve the equations having 6 2D and 3D points to get an initially estimated camera pose. PnP is prone to error as there are outliers in the given set of point correspondences. To overcome this error, we can use RANSAC to make our camera pose more robust to outliers. To formalize, given N¿6, 3D-2D correspondences, implement the following function that estimates camera pose (C, R) via RANSAC. Similarly to non-linear triangulation, we refine the output from LinearPnP in Non-linear PnP by minimizing the reprojection error.

F. Bundle Adjustment

We are one step closer to having the 3D reconstructed output of the scene now that we have the set of refined 3D points from various perspectives and camera poses. Bundle adjustment refines both camera poses and 3D points at the same time by minimizing reprojection error. It simultaneously refines the 3D coordinates describing the scene geometry, relative motion parameters, and optical characteristics of the cameras used to acquire the images, using an optimality criterion involving the corresponding image projections of all points.

The visibility matrix is a Boolean matrix that denotes if a particular feature is visible in a particular image. In bundle adjustment, we create a sparse matrix that records whether a 2D point observation belongs to a particular parameter.

II. Phase 2: NeRF

In this section, we implement the Neural Radiance Fields(NeRF) paper. It is a method that synthesizes novel views of complex scenes given finite input images to the network. It optimizes a continuous volumetric scene function and achieves state-of-the-art results.

A. Data Loading

For loading the data, we have taken the data from the original authors' link which has 3 sets of 100 images for each of train, validation and test. We have used the provided JSON files for each set to get the transformation matrices for each camera pose. The focal length is taken to be 138.88887. The images and transformation matrices are basically lists that are converted to torch tensors and returned.

B. Ray Generation and Query Points

We generate the rays that start from the camera centers and pass through each pixel of the image acquired from that particular camera. The rays are then sampled at a certain number of points (32 or 64) to get points on each ray. These points are called query points. These points are in the form of a torch tensor. These query points are then flattened by converting them into a vector of dimension (N,1). The flattened points are encoded by an encoding function before passing them to the model.

C. Model Architecture

The model architecture is the same as the one used by the authors and given in the paper. It is an MLP model with 8 fully connected layers each one activated by the ReLU function. It features skip connections after every 4 layers where the output is concatenated with the input before giving to the next layer. The 8th fully connected layer has no activation while the last fully connected layer(10th) uses a sigmoid activation to output the RGB map. The architecture is shown in the figure below. The loss function used is the MSE loss and the optimizer is ADAM. The learning rate is taken to be 5e-4.

D. Radiance field and Volume Rendering

The model output is un-flattened to obtain the radiance field. Then we perform volume rendering which then re-synthesizes the RGB image. It takes input as the radiance field and renders

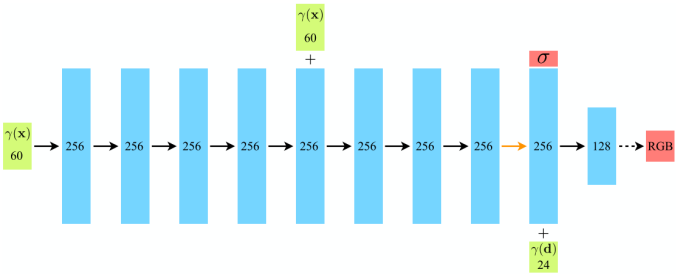

Fig. 4: Detected corners (Custom Set 1)

it given the origin of each ray in the input group of rays(chunk) and the sampled values along with them. Volume rendering also gives the depth map and the accumulated transmittance map but we are only using the RGB map.

E. Results

We run our model for 1000 iterations and we are saving the rendered RGB image which is the model output every 100 iterations so that we have a brief idea of how the rendered image improves over time. These 10 output images are shown below.

F. Improvements that can be made and observations

We observed that changing the percent of inliers affects the RANSAC calcultaions and in-turn the stitching. So to improve the accuracy for stitching we increased the probability of outliers in every iteration. We also decreased the number of corners that are generated iteratively. When the images go beyond a threshold number of 3 this process is triggered. This ensures that the best matches are retained and the homography matrix calculated is proper.

We also wanted to test another approach where we separate the number of images into three parts and stitch them together at the end and also dynamically select matches from only specific regions of the images. This would probably lessen the problems while stitching and blending. However, due to time constraints, this approach was not implemented.

G. Results

Following are sample results for Test Set1 and Custom Set1. Other results are shown at the end of the report. The results from all the panoramas are shown in Fig (33).

H. Special thanks to -

https://www.cse.psu.edu/ rtc12/CSE486/lecture12.pdf
https://www.cis.upenn.edu/ cis580/Spring2016/Lectures/cis580-18-coursera-2016-SfM-fulll.pdf
https://www.cis.upenn.edu/ cis580/Spring2015/Projects/proj2/proj2.pdf
https://github.com/hal2001/Camera-Rectification-and-Structure-from-Motion

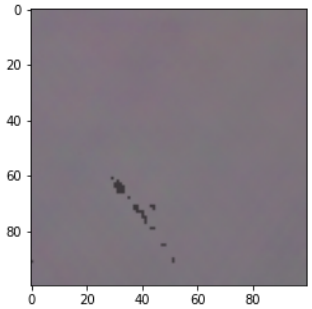

Fig. 5: Iteration - 100

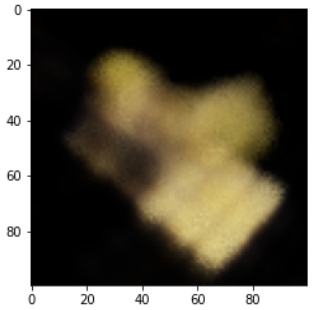

Fig. 6: Iteration - 200

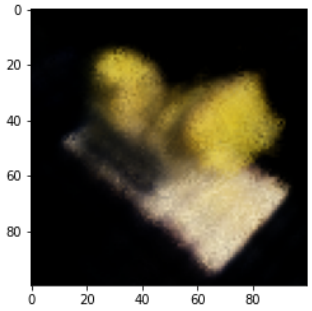

Fig. 7: Iteration - 300

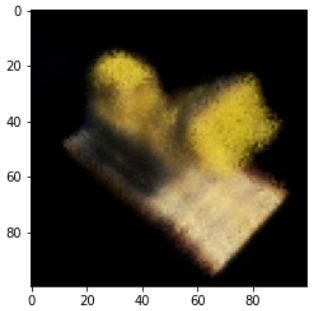

Fig. 8: Iteration - 400

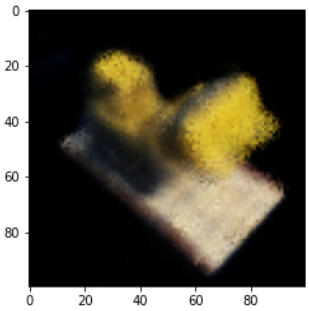

Fig. 9: Iteration - 500

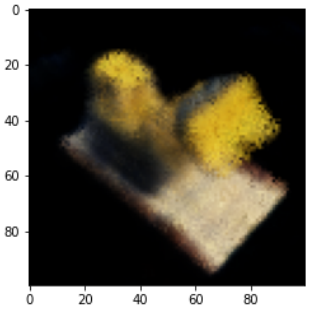

Fig. 10: Iteration - 600

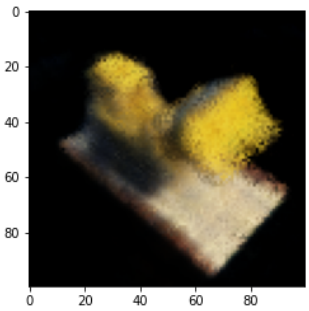

Fig. 11: Iteration - 700

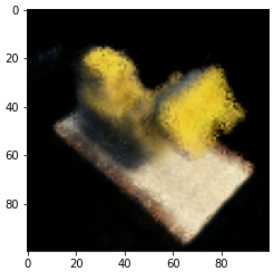

Fig. 12: Iteration - 800

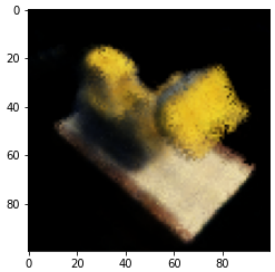

Fig. 13: Iteration - 900

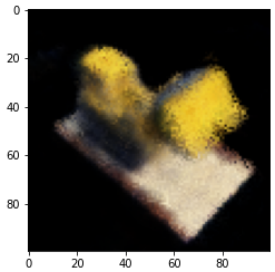

Fig. 14: Iteration - 1000