Structure from Motion

Ning Cao
cao.nin@northeastern.edu

Abstract

This report presents the implementation and results of the Structure from Motion (SfM) algorithm using the factorization method on the CMU Hotel Sequence dataset. The algorithm involves feature extraction, tracking across frames, and the recovery of the 3D structure. We discuss the challenges faced, optimizations made, and possible improvements to the algorithm.

1. Introduction

Structure from Motion (SfM) is a technique for recovering the 3D structure of a scene from a sequence of 2D images taken from different viewpoints. The factorization method is an approach to solving the SfM problem by decomposing a measurement matrix into camera motion and structure matrices. In this project, we implemented the SfM algorithm using the factorization method and tested it on the CMU Hotel Sequence dataset.

Code for our project can be found in our Github (....), linked in the Appendix.

2. Approach

2.1. Feature extraction

We used the Scale-Invariant Feature Transform (SIFT) algorithm from the OpenCV library to extract features from each image in the sequence. SIFT is an algorithm that detects and describes local features in images. It is robust against changes in scale, rotation, and illumination. SIFT extracts keypoints and computes their corresponding feature descriptors, which are 128-dimensional vectors representing the local image gradient information around each keypoint. The result is shown in Figure 1.

2.2. Feature tracking

We tracked the features across the frames using the Lucas-Kanade Optical Flow method with the calcOpticalFlowPyrLK function in the OpenCV library. The Lucas-Kanade method is a differential method for estimating the optical flow between two images, relying on the brightness constancy assumption. It is a local method, and we applied it to the keypoints detected in each image.

The calcOpticalFlowPyrLK function estimates the optical flow for a sparse feature set using the iterative Lucas-Kanade method with pyramids. It takes several parameters, such as the previous and next image, previous keypoints, and window size, among others. The output of the function is the next set of keypoints and a status array that indicates if the corresponding keypoint was successfully tracked. The result is shown in Figure 2.

2.3. Factorization method

We constructed a measurement matrix W using the tracked feature points. Each column of W corresponds to a feature point tracked across the frames, and each row pair represents the x and y coordinates of the feature points in a particular frame.

After constructing the measurement matrix W, we applied Singular Value Decomposition (SVD) to factorize it into camera motion (M) and structure (S) matrices:

$$W = M * S$$

SVD decomposes W into three matrices U, S, and Vt:

$$W = U * np.diag(S) * Vt$$

We selected the first three columns of U and the first three rows of Vt, and the first three singular values in S to form the matrices M and S. The structure matrix S contains the 3D coordinates of the points:

$$M = U[:, :3] * np.sqrt(np.diag(S[:3]))$$

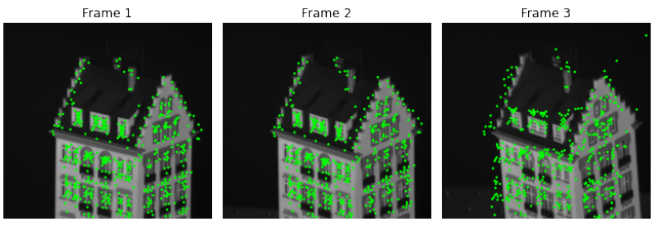
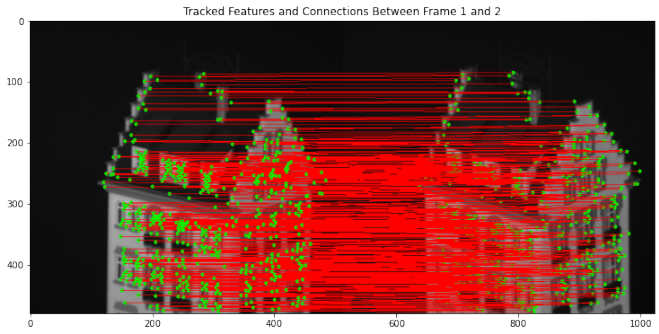
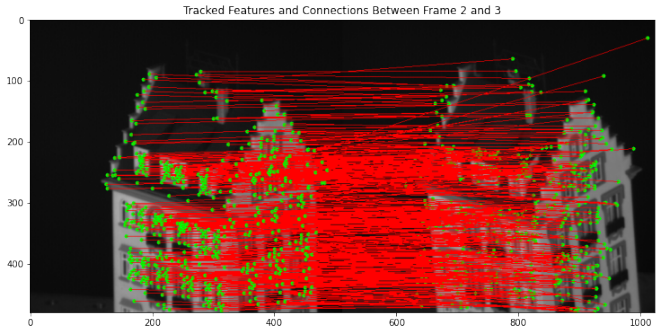


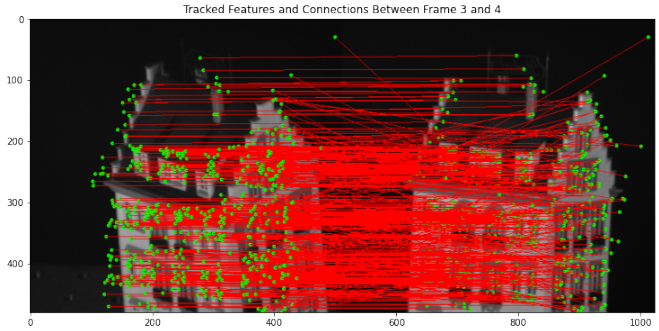
Figure 1. Features across frames



(a) Tracked Features and Connections Between Frame 1 and 2



(b) Tracked Features and Connections Between Frame 2 and 3



(c) Tracked Features and Connections Between Frame 3 and 4

Figure 2

$$S = np.sqrt(np.diag(S[:3])) * Vt[:3,:]$$

The result is shown in Figure 3.

2.4. PLY file generation

We saved the 3D coordinates in an ASCII output PLY file format to visualize the results using Meshlab. The PLY file format is a simple and widely-used format to store 3D point cloud data. It can store additional information like color, normals, and texture coordinates. However, for our purpose, we only stored the 3D coordinates (x, y, z) of each point.

We created a function $save_plyfile$ that takes the 3D coordinates and the output file name as input. The function uses the plyfile library to create a PlyData object with the 3D coordinates and writes it to the specified output file in ASCII format. This PLY file can then be opened and visualized using Meshlab or other compatible 3D viewers.

3. Conclusions

In this report, we presented the implementation and results of the Structure from Motion algorithm using the factorization method on the CMU Hotel Sequence dataset. Despite some challenges, the algorithm was able to recover the 3D structure of the scene. Further improvements and optimizations can be made to enhance the performance and robustness of the algorithm.

4. Appendix: Code

Please see our git repository for copies of our code. https://github.com/vegetablesB/EECE5639Project

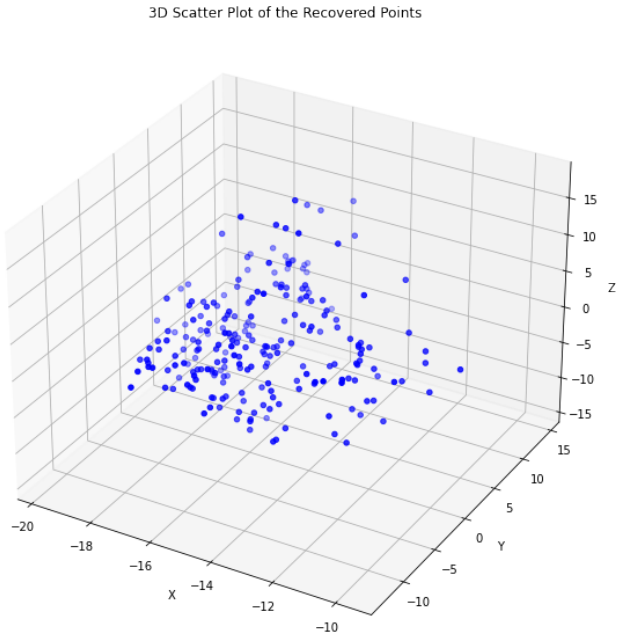


Figure 3. 3D Scatter Plot of the Recovered Points