Machine Perception
Project 2 Report

"Shape from Motion"

Hongyi Fan

Yifan Yin

Table of Contents

Objective

For this project, we used the Tomasi-Kanade algorithm to reconstruct the shape of an object from a sequence of images. Two sets of images were used in the experiment: the first set consisted of 28 consecutive images of a castle taken from a moving camera, and the second set consisted of pictures taken from around a Medusa statue face. The project aimed to demonstrate the Tomasi-Kanade algorithm's ability to approximately reconstruct an object's shape from a series of images.

Figure 1: The castle(left) and the face of Medusa(right)
Here only showing one frame among the sequence

Approach

Feature Extractor

We used the “Good Features to Track”[1] implementation in the OpenCV library to extract features from the image frames. This algorithm is proposed by Shi-Tomasi in Proceedings of the 1994 IEEE Conference on Computer Vision and Pattern Recognition. It is widely associated with the Tomasi-Kanade algorithm.

Optical Flow

The optical flow algorithm we are using to track the features extracted is the Lucas-Kanade method [2].

A 3x3 patch is used in the Lucas-Kanade method around the point. Thus, the motion of all nine points is the same. For these nine points, we can locate (fx, fy, ft). As a result, solving nine equations with two unknown variables that are overdetermined is now our challenge.

The least-square fit method yields a superior solution. The final solution, which consists of solving two unknown problems and two equations, can be found below.

$$\begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} \sum_i {f_{x_i}}^2 & \sum_i f_{x_i} f_{y_i} \\ \sum_i f_{x_i} f_{y_i} & \sum_i {f_{y_i}}^2 \end{bmatrix}^{-1} \begin{bmatrix} -\sum_i f_{x_i} f_{t_i} \\ -\sum_i f_{y_i} f_{t_i} \end{bmatrix}$$

In the implementation, we either use the given frame datasets (in the case of the castle), or use a frame reader object from OpenCV to capture the frames (in the case of medusa statue), one at each time. In the very first frame, we call the “Good Features to Track” extractor to extract a given amount of features. As each frame follows, we track the same combination of features in that frame by using the implementation `cv.calcOpticalFlowPyrLK()`. This function takes two adjacent frames as well as features p0 from the old frame, and returns the same feature points in the new frame and a status vector. If the feature is found in the new frame, status of that frame is set to be 1, otherwise it will be set to 0.
After reading all the frames, we only keep those features that live in all frames using the status vectors returned.

Tomasi-Kanade Factorization[3]

Let’s assume there are *F* frames of images in the sequence. And n is the number of feature points tracked throughout the sequence. Each tracked point is a 2D vector representing its position in image coordinates.
We can construct matrices *U* and *V*. Where U contains the x coordinates of the feature points and V contains the y coordinates of the feature points. Both *U* and *V* have a size of $F \times n$.
Having the information from above, we construct the observation matrix *W* by stacking *U* and *V* together.

$$W = [\frac{U}{V}]$$

Apply Singular Value Decomposition to the observation matrix W

$$W = ADB^T$$

Ideally, there should be three non-zero singular values. We only take the first 3 columns of A and the first 3 rows of B to ignore singular values caused by noise.

The camera motion matrix M can be calculated by

$$M = A \times D$$

The structure matrix X can be calculated by

$$X = B^T$$

For the purpose of reconstruction, we only need matrix X.

3D Reconstruction and Visualization

The shape matrix *S* contains all the information needed for the 3D reconstruction. Matrix *S* takes a size of 3*n, each column of which corresponds to one constructed 3D point in space. Plot all the points obtained, and we end up with a point cloud, which is our reconstruction result.

In the implementation, we use the function `ax.scatter()` in Python library *matplotlib.pyplot* for the visualization.

Result and Analysis

Castle

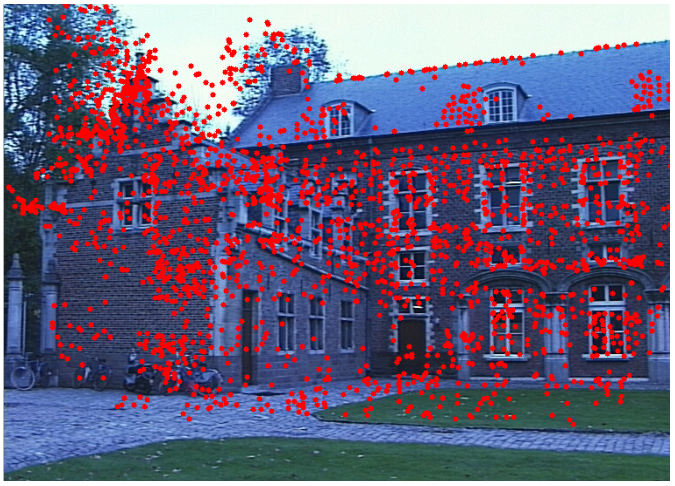

Figure 2: Features extracted from the Castle

We initially commanded 3000 feature points. After the elimination of the optical flow tracking, we end up having 2286 points remaining.

Figure 3: The optical flow draws on the last image frame

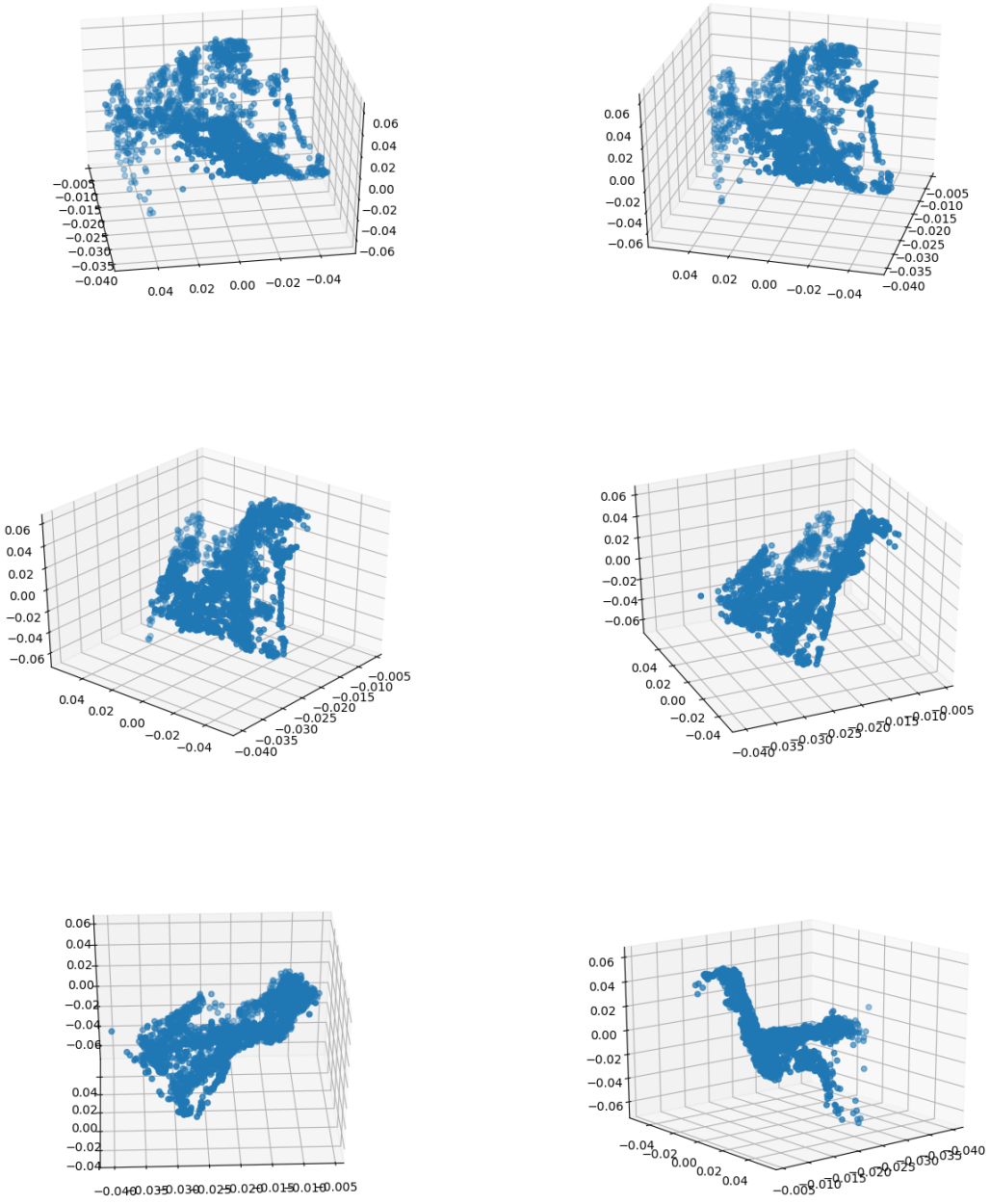

Figure 4: Results of 3D Reconstruction of the castle

Medusa Statue

In this experiment, 5000 features with a minimum distance of 1 pixel are extracted at the very first frame. After going through all the frames, in total 3803 common features are tracked throughout the analysis. The reconstrued statue is plotted below.

Figure 5: Results of 3D Reconstruction of the Medusa statue

Discussion

Performace

From the images and analysis obtained, we conclude that the algorithm is well performed, and the 3D reconstruction results give decent recoveries of the original objects in 3D space. In the castle case, we can tell the windows, roofs and trees roughly from the point cloud. In the Medusa statue case, the selected features are roughly around a plane in space, which corresponds to the face of Medusa status in the source video.
It takes less than 1 minute (53s) to run 5000 features points, which is an indicator that the algorithm is fairly good in terms of time complexity.

Strengths and Drawbacks

The obvious drawback is that our implementation does not have a quantitative metric to measure its performance. What we have concluded is based on human visual intuition. To further improve this implementation, we need to include a metric for evaluating the algorithm's performance.
The other drawback is that occluded points are eliminated during the process of optical flow computation. The features that lost track between frames are discarded and do not participate in the structure computation at the end. However, there are methods to utilize the occluded points in the factorization.

Reference

[1]Jianbo Shi and Tomasi, "Good features to track," 1994 Proceedings of IEEE Conference on Computer Vision and Pattern Recognition, 1994, pp. 593-600, doi: 10.1109/CVPR.1994.323794.
[2] Lucas, Bruce D., and Takeo Kanade. An iterative image registration technique with an application to stereo vision. Vol. 81. 1981.