ELEC5630 (L1) Project 3

Structure from Motion

Professor: TAN, Ping

Due Nov. 9, 2024

1 Structure from Motion

In this project, you will reconstruct a 3D point cloud and camera pose of a given image set. Your task is to implement the full pipeline of structure from motion, including two-view reconstruction, triangulation, PnP, and bundle adjustment. In this project, you can use high-level functions provided by Opencv, such as findEssentialMatrix. You can choose an optimizer such as built-in functions in Python package for nonlinear optimization parts.

In this project, you should implement the code by yourself. Every script and function you write in this section should be included in the *python/* directory. The file "main.py" will be run as the program entry to evaluate the result.

There are two datasets used in this project. The simple one is called "templeRing"; you can find it in the zip file. Another one will be the *llff*, which can be downloaded from here. You only need to reconstruct the "fern" and "trex" scenes. You can also find the sample to read the data in at here. This project assumes the intrinsic matrix and distortion parameters are KNOWN. Thus, we can directly read these parameters from the dataset. You can not use the extrinsic matrix, but you should evaluate the accuracy of the pose generated by your algorithm with the ground truth. You should report the RMSE for both rotation and translation on the LLFF dataset.

This project is harder than the previous projects. You might fail to reconstruct both datasets. You will get 20% points for each successful reconstruction (20% for "templering", 20% for two scenes on LLFF). If you fail to reconstruct LLFF(not for "templering"), you can submit a report to analyze why the reconstruction fails to get full mark for LLFF.

Post questions to Canvas so everybody can share unless the questions are private. Please look at Canvas first if similar questions have been posted.

2 Method

2.1 Overview

The full pipeline of structure from motion is shown in Algorithm 1. You will program this full pipeline guided by the functions described in the following sections. This pseudocode does not include data management, e.g., converting matches obtained from the data files to feature points.

Algorithm 1: Structure from Motion

```
Input  : $N$ Image set $\{I_i\}$ with intrinsic matrix $\{K_i\}$
Output: 3D positions of points $\{X_i\}$ and camera poses $\{P_i\}$
Initialization:
1  $[x_0, x_1]$ = FindCorrespondence($I_0, I_1$) // Find the correspondence in first two images.
2  $E_{01}$ = FindEssentialMtrix($x_0, x_1, K_0, K_1$) // Computing essential matrix.
3  $[R_{01}, t_{01}]$ = RecoverPoseFromEssential($E_{01}, x_0, x_1$) // Recovering Pose.
4  $X_{all}$ = Triangulation($K, R_{01}, t_{01}, x_0, x_1$)
Incremental Optimization:
5  $i \leftarrow 2$
   while ($i < N$) do
6      $x = [x_j, x_i]$ = FindCorrespondence($I_j, I_i$) // $j$ is previous frame.
7      $[R_i, t_i]$ = PnP($X, x, K_i$) // Resgiter the $i^{th}$ image.
8      $X_{all} \bigcup$ = Triangulation($K, R_{ij}, t_{ij}, x_i, x_j$) // Adding new points
9      $V$ = BuildVisibilityMatrix($PointTraj$) // visibility mtraix for BA
10     $[R, t, X_{all}]$ = BundleAdjustment($R, t, X_all, K, V$) // Bundle adjestment.
   end
```

2.2 Matching

2.2.1 Correspondence

You can use open-source non-commerce packages to detect feature points and find matching, such as functions in opencv, superglue, or lightglue. There is no limitation in this project.

2.2.2 Fundamental Matrix Estimation

Given $N \geq 8$ correspondences between two images, you can linearly estimate a fundamental matrix $\mathbf{F}$, which minimizing $x_2^\top \mathbf{F} x_1$.

The fundamental matrix can be estimated by solving linear least squares (Ax = 0). Because of noise on correspondences, the estimated fundamental matrix can be ranked 3. The last singular value of the estimated fundamental matrix must be set to zero to enforce the rank 2 constraint.

Another approach to eliminating the outlier is RANSAC, which is widely applied in OpenCV.

You can use the function in opencv to compute the fundamental matrix.

2.3 Relative Pose Estimation

2.3.1 Essential Matrix Estimation

Given $\mathbf{F}$, you decompose $\mathbf{E}$ from $\mathbf{F}$ by $\mathbf{E} = \mathbf{K}^\top \mathbf{F} \mathbf{K}$.

An essential matrix can be extracted from a fundamental matrix given camera intrinsic parameter, $\mathbf{K}$. Due to noise in the intrinsic parameters, the singular values of the essential matrix are not equal(1,1,0). The essential matrix can be corrected by reconstructing it with (1,1,0) singular values.

You can also compute the essential matrix by a five-point algorithm provided in OpenCV or extract it from the fundamental matrix.

2.3.2 Camera Pose Extraction

Given $\mathbf{E}$, you can decompose 4 camera pose configurations. You can find figure 10.2 on Page 265 of *Multiview Geometry in Computer Vision.* Usually, the correct one can be selected with the triangulation points. We will select the one under *cheirality* condition, i.e. the reconstructed points must be in front of the cameras. To check the cheirality condition, triangulate the 3D points (given two camera poses) using linear least squares to check the sign of the depth in the camera coordinate system w.r.t. camera center..

You can use the function in opencv to recover the camera pose.

2.4 Triangulation

Given the correspondence and the projective matrix, the 3D point can be computed by triangulation via DLT algorithm.

You can use the function in opencv to recover the camera pose for triangulation.

2.5 Perspective-n-Point

Given 2D-3D correspondences and the intrinsic parameter, estimate a camera pose using linear least squares. You can find the material on the PnP in the course slides.

You can use the function in OpenCV to recover the camera pose for triangulation.

2.6 Bundle Adjustment

Given initialized camera poses and the 3D points, refine them by minimizing reprojection error.

In Python, you can use the function provided in numpy, scipy or other Python packages. You can also use **fminunc** or **lsqnonlin** to do so.

2.7 Non-linear Optimization (Optional)

In sections PnP and triangulation, most algorithms are computed by solving linear equations, which minimizes algebra error. However, it is not enough to perform well for the final reconstruction. A solution is deploying a non-linear optimization after each component to further refine the output of the linear solver.

The linear triangulation minimizes algebraic error. Geometrically meaningful reprojection error is computed by measuring the error between the measurement and the projected 3D point, similar to bundle adjustment. You can deploy a non-linear optimization after each triangulation to minimize projection error for a better result.

Similar with BA and non-linear triangulation, the non-linear PnP also minimizes the projection error. You can deploy a non-linear optimization after each PnP to minimize projection error for a better result.

This part will provide an extra 20% bonus points for the final grade.

3 Submission

You only need to upload **ONE** zip file containing the code, the README file, and the results. **Please do not upload the full dataset. You can prepare an empty folder called *"/data"* as a placeholder.** The code should be in Python format. Implementing another language will be skipped and get 0 pts in this project.

You are allowed to use the high-level function provided in Python. If you implement functions by yourself, you can gain an extra 20% bonus point on the final grade.

If you implement the non-linear optimization for PnP and triangulation, you can gain an extra 20% bonus point on the final grade.

You should output the computed camera poses and compare them with the ground truth. You should also report RMSE of rotation and translation.

You should upload a 3D visualization of the 3D points and the cameras. It can be a sequence of images or a video.

The quality of the final 3D model reconstruction does not affect the scoring. If you cannot reconstruct the final result, you need to submit a report to analyze why the reconstruction failed.

You should write the necessary information in the README file, including the environment, the cmd, the file format, etc. You should make the code in the submission self-contained. The script output should be matched with the file in the results folder.

Cite the paper, GitHub repo, or code url if you use or reference the code online. Please keep academic integrity; plagiarism is not tolerated in this course.

4 Tips

You can use evo or other packages to compute the RMSE of the tranlation.

You can use python packages like, cv2, numpy, to finish this project.