Iterative Factorization Method for Visual SLAM

A project of the 2019 Robotics Course of the School of Information Science and Technology (SIST) of ShanghaiTech University
https://robotics.shanghaitech.edu.cn/teaching/robotics2019

Zhouyang Lin, Guoliang Gao

January 8, 2020

Abstract

Factorization method is a classical algorithm to calculate transformation and recover 3D shape of objects. It is applicable when the objects are far away from the camera based on orthogonal camera model. In this project, we propose to improve the current factorization method with iterative process to increase the accuracy of localization. And to achieve the reconstruction of the earth model

Keywords: Factorization, 3D shape, iteration

1 Introduction

'Shape from motion' exploits the relative motion between camera and scene. Similar to the stereo technique the process can be divided into the subprocesses finding of correspondence from consecutive frames and reconstruction of the scene. The differences between consecutive frames are, on average, much smaller than those of typical stereo pairs, because image sequences are sampled at high rates.

Regarding correspondence, the fact that motion sequences provide many closely sampled frames for analysis is an advantage. Differential methods use estimates of time derivatives and require therefore image sequences sampled closely. This method is computed at each image pixel and leads to dense measurements. Matching methods use Kalman filtering to match and track efficiently sparse image features over time. This method is computed only at a subset of image points and produces sparse measurements.

An image stream can be represented by the 2FxP measurement matrix of the image coordinates of P points tracked through F frames. We show that under orthographic projection this matrix is of rank 3. Based on this observation, the factorization method uses the singular value decomposition technique to factor the measurement matrix into two matrices which represent object shape and camera rotation respectively. Two of the three translation components are computed in a preprocessing stage. Then we can construct the new matrix W that after finish tracking the feature points in every image frames with the size of 2FxP. After got the updated matrix, using SVD decomposition to calculate the rotation matrix R and the shape matrix S. The matrix S is the result we want to show. The method can also handle and obtain a full solution from a partially filled-in measurement

matrix that may result from conclusions or tracking failures. The method gives accurate results, and does not introduce smoothing in either shape or motion.

In this project, we mainly want to focus on application of satellite, which is suitable for orthogonal projection. Some datasets of physical simulation have been collected. However, the performance of localization and 3D reconstruction may not meet the standard. Therefore, it is needed to improve the algorithm with iterative method.

We plan to implement the original factorization method in C++ and improve it with the core idea of iterative methods. Besides, we will test the algorithm on the simulation datasets. If the results is good enough, we will try to test it on real images captured by satellite and submit a paper about it.

2 State of the Art

2.1 Member one

'Scene understanding using DSO Cognitive Architecture'presented the DSO-CA is able to bring to bear different types of knowledge to solve problems. The cognitive architecture is able to fuse bottom-up perceptual information, top-down contextual knowledge and visual feedback in a way similar to how human utilizes different knowledge to recognize objects in images or video scene. This enables the cognitive system to achieve scene understanding.

'SVO in array diagnostic for the planar near-field scanning'put forward a method for the array characterization from Near-Field measurements is here presented. The method exploit the concept of Singular-Value Optimization to dramatically reduce the number of data needed by a complete mapping of the excitations. The approach is here applied to a planar scanning and numerically validated.

'ORB-SLAM-based Tracing and 3D Reconstruction for Robot using Kinect 2.0'provide dense 3D reconstruction while exploring environment in real time, based on one of the best SLAM algorithm called ORB-SLAM, which accurately estimate the camera poses and sparse 3D map of the environment based on images.

There are more details in this paper.ORB extract feature points based on oriented FAST with direction, and binary descriptors are used to improve the speed, which has good scale invariance and noise suppression performance[20]. The system uses the ORB algorithm to detect the keypoints and calculate the descriptors of keypoints. When the tracking of the previous frame is successful, the initial position of the camera is estimated from the previous frame. Each keypoint of the previous frame and its descriptor has its associated point, the matched point is searched in the small region near the corresponding location of the current frame. The initial corresponding point is optimized by the direction consistency check. At this time, a series of correspondence is obtained, so PnP solution can be solved in the RANSAC scheme to calculate the initial camera pose. When the tracking of the previous frame fails, the current frame is described by the visual bag model and is searched for in the visual word bag database made up of key frames to realize global relocation. The initial pose estimation obtained by inter-frame registration has accumulated errors, and the current pose can be further optimized using the constructed map. The keypoints of the previous keyframes are projected into current frame, and the residual error can't be overlapped with the corresponding keypoints to construct the residual of the estimated position. The position of the keypoints obtained after optimization is theoretically more accurate than the previous estimation of the camera pose, and thus the camera pose and the position of the keypoints in the map can be optimized. Increasing the frame rate of the camera will improve accuracy and robustness. However, maintaining the data of all the images requires a lot of resources but serious redundancy. In order to track the movement

of the camera more robustly, it determines whether the current frame can be a new key frame and insert the appropriate keyframe.

As for the ROS package, Open3D is an open-source library that supports rapid development of software that deals with 3D data. It frontend exposes a set of carefully selected data structures and algorithms in both C++ and Python. The backend is highly optimized and is set up for parallelization. Open3D was developed from a clean slate with a small and carefully considered set of dependencies. It can be set up on different platforms and compiled from source with minimal effort. The code is clean, consistently styled, and maintained via a clear code review mechanism. Open3D has been used in a number of published research projects and is actively deployed in the cloud. Open3D provides data structures for three kinds of representations: point clouds, meshes, and RGB-D images. For each representation, we have implemented a complete set of basic processing algorithms such as I/O, sampling, visualization, and data conversion. In addition, we have implemented a collection of widely used algorithms, such as normal estimation, ICP registration, and volumetric integration. The most useful for our project is that a sophisticated workflow that is demonstrated in an Open3D tutorial is a complete scene reconstruction system. The system is implemented as a Python script that uses many algorithms implemented in Open3D. It takes an RGB-D sequence as input and proceeds through three major steps:
1. Build local geometric surfaces Pi (referred to as fragments) from short subsequences of the input RGB-D sequence. There are three substeps: matching pairs of RGB-D images, robust pose graph optimization, and volumetric integration.
2. Globally align the fragments to obtain fragment posesTi and a camera calibration function C(·). There are four substeps: global registration between fragment pairs, robust pose graph optimization, ICP registration, and global non-rigid alignment.
3. Integrate RGB-D images to generate a mesh model for the scene.

2.2 Member two

2.2.1 Section one

[**?**] developed a factorization method that can overcome the difficulty occurring when the objects are distant with respect to their size. by recovering shape and motion under orthography without computing depth as an intermediate step.

[**?**] describes both batch and online shape-from-motion algorithms for omnidirectional cameras, and a precise calibration technique that improves the accuracy of both methods.

[**?**] introduced the factorization methods of recovering the 3D structure and motion of an object from image sequence systematically and detailedly, from the point of view on rigid object and nonrigid object. Merits and drawbacks of the techniques mentioned are analyzed and compared. Finally, some problems of current theories and methods are proposed, the problems which need further research and the development trend of this domain are presented.

2.2.2 Section two

Shape-from-motion is common knowledge that existing solutions work well for perfect images, but are very sensitive to noise. This paper presented a new method called thefactorization method which could robustly recover shape and motion from a sequence of images under orthographic projection. The effects of camera translation along the optical axis were not accounted for by orthography. Consequently, this component of motion could not be recovered and must be small relative to the scene distance. However, this restriction to shallow motion improved dramatically the quality of the computed shape and of the remaining five motion parameters. This paper demonstrated this with a series of experiments on laboratory and outdoor sequences, with and without occlusions.

In the factorization method, this paper represented an image sequence as a $2F \times P$ measurement matrix W, which was made up of the horizontal and vertical coordinates of P points tracked through F frames. If image coordinates were measured with respect to their centroid, this paper prove the rank theorem: under orthography, the measurement matrix is of rank 3. As a consequence of this theorem, it showed that the measurement matrix could be factored into the product of two matrixes R and S. Here, R was a $2F \times 3$ matrix that represents camera rotation, and S was a $3 \times P$ matrix that represented shape in a coordinate system attached to the object centroid. The two components of the camera translation along the image plane were computed as averages of the rows of W. When features appeared and disappeared in the image sequence because of occlusions or tracking failures, the resulting measurement matrix W was only partially filled in. The factorization method could handle this situation by growing a partial solution obtained from an initial full submatrix into a complete solution with an iterative procedure.

2.2.3 Section three

RPG_TRAJECTORY_EVALUATION implements common used trajectory evaluation methods for visual(-inertial) odometry. Specifically, it includes different trajectory alignment methods (rigid-body, similarity and yaw-only rotation). Commonly used error metrics: Absolute Trajectory Error (ATE) and Relative/Odometry Error (RE).

Since trajectory evaluation involves many details, the toolbox is designed for easy use. It can be used to analyze a single trajectory estimate, as well as compare different algorithms on many datasets (e.g., this paper) with one command. The user only needs to provide the groundtruths and estimates of desired format and specify the trajectory alignment method. The toolbox generates (almost) paper-ready plots and tables. In addition, the evaluation can be easily customized.

3 System Description

First we use an autonomous robot to collect the picture of the earth model as the dataset. Then use Harris operator to calibrate the feature points. Opticalflow method can track these feature points' trajectory. Then we can use the following method to rebuid the model. Suppose that we have tracked P feature points over F frames in an image stream. We then obtain trajectories of image coordinates $\{(u_{fp}, v_{fp}) \mid f = 1, ..., F, p = 1, ..., P\}$. We write the horizontal feature coordinates u_{fp} into an $F \times P$ matrix U with one row per frame and one column per feature point. Similarly, an $F \times P$ matrix V is built from the vertical coordinates v_{fp}. The combined matrix of size $2F \times P$

$$W = [\frac{U}{V}] \tag{1}$$

is called the measurement matrix. The rows of the matrices U and V are then registered by subtracting from each entry the mean of the entries in the same row:

$$\begin{aligned} \tilde{u}_{fp} &= u_{fp} - a_f \\ \tilde{v}_{fp} &= v_{fp} - b_f \end{aligned} \tag{2}$$

where

$$\begin{aligned} a_f &= \frac{1}{P} \sum_{p=1}^{P} u_{fp} \\ b_f &= \frac{1}{P} \sum_{p=1}^{P} v_{fp} \end{aligned} \tag{3}$$

This produces two new $F \times P$ matrices $\tilde{U} = [\tilde{u}_f p]$ and $\tilde{V} = [\tilde{v}_f p]$. The new matrix

$$\tilde{W} = [\frac{\tilde{U}}{\tilde{V}}] \tag{4}$$

is called the registered measurement matrix. This is the input to our factorization method.

Suppose that we place the origin of the world reference system at the centroid of the P points $s_p = (x_p, y_p, z_p)^T, p = 1, ..., P$ in space that correspond to the P feature points tracked in the image stream. The orientation of the camera reference system corresponding to frame number f is determined by a pair of unit vectors i_f, j_f pointing along the scanlines and the columns of the image respectively, and defined with respect to the world reference system. Under orthography, all projection rays are then parallel to the cross product of i_f and j_f:

$$k_f = i_f \times j_f \tag{5}$$

The projection (u_{fp}, v_{fp}) (the image feature position) of point $s_p = (x_p, y_p, z_p)^T$ onto frame f is given by equations

$$\begin{aligned} u_{fp} &= i_f^T (s_p - t_f) \\ v_{fp} &= j_f^T (s_p - t_f) \end{aligned} \tag{6}$$

where $q = (a_f, b_f, c_f)^T$ is the vector from the world origin to the origin of image frame f.

Note that since the origin of the world coordinates is placed at the centroid of the object points, we have

$$\frac{1}{P} = \sum_{p=1}^{P} s_p = 0 \tag{7}$$

We can now write expressions for the entries $\tilde{u}_{fp}$, and $\tilde{v}_{fp}$ defined in (2) of the registered measurement matrix. For the registered horizontal image projection we have

$$\begin{aligned}\tilde{u}_{fp} &= u_{fp} - a_f \\ &= i_F^T(s_p - t_f) - \frac{1}{P}\sum_{q=1}^{P} i_f^T(s_q - t_f) \\ &= i_f^T(s_p - \frac{1}{P}\sum_{q=1}^{P} s_q) \\ &= i_f^T s_p\end{aligned} \tag{8}$$

We can write a similar equation for $\tilde{v}_{fp}$. To summarize,

$$\begin{aligned}\tilde{u}_{fp} &= i_f^T s_p \\ \tilde{v}_{fp} &= j_f^T s_p\end{aligned} \tag{9}$$

By collecting the two sets of $F \times P$ equations (9), the registered measurement matrix $\tilde{W}$ (equation (1)) can be expressed in a matrix form:

$$\tilde{W} = RS \tag{10}$$

where

$$R = [i_1^T, ..., i_F^T, j_1^T, ..., j_F^T]' \tag{11}$$

represents the camera rotation and

$$S = [s_1, ..., s_p] \tag{12}$$

is the shape matrix. In fact, the rows of R represent the orientations of the horizontal and vertical camera reference axes throughout the stream, while the colunms of S are the three-dimensional coordinates of the P feature points with respect to their centroid.

The rank theorem expresses the fact that the $2F \times P$ image measurements are highly redundant. Indeed, they could all be described concisely by giving F frame reference systems and P point coordinate vectors, if only these were known.

From the first and the last line of equation (8), the original unregistered matrix W can be written as

$$W = RS + te_p^T \tag{13}$$

where $t = (a_1, ..., a_F, ..., b_f)$ is a $2F$-dimensional vector that collects the projections of camera translation along the image plane (see equation (8)), and $e_p^T = (1, ..., 1)$ is a vector of P ones. In scalar form,

$$\begin{aligned}u_{fp} &= i_f^T s_p + a_f \\ v_{fp} &= j_f^T s_p + b_f\end{aligned} \tag{14}$$

Comparing with equations (9), we see that the two components of camera translation along the image plane are simply the averages of the rows of W.

In the equations above, i_f and j_f are mutually orthogonal unit vectors, so they must satisfy the constraints

$$|i_f| = |j_f| = 1 \; and \; i_f^T j_f = 0 \tag{15}$$

Also, the rotation matrix R is unique if the system of reference for the solution is aligned, say, with that of the first camera position, so that

$$i_1 = (1, 0, 0)^T \; and \; j_i = (0, 1, 0)^T \tag{16}$$

4 System Evaluation

We want to test our system with [**?**] and [**?**]. In detail, we would like to use trajectory alignment methods to calculate absolute trajectory error (ATE) and relative odometry error (RE).

For a success system, it should meet these three requirements:

$$\begin{aligned} |pe| &< 1 \\ |ye| &< 1 \\ |re| &< 0.5 \\ |rpe| &< 0.1 \\ |rye| &< 0.1 \\ |rre| &< 0.05 \end{aligned} \tag{17}$$

Here $|pe|$, $|ye|$, $|re|$, $|rpe|$, $|rye|$, $|rre|$ represent the absolute values of pitch error, yaw error, roll error, relative pitch error, relative yaw error, relative roll error respectively.

[**?**] implements common used trajectory evaluation methods for visual(-inertial) odometry. Since trajectory evaluation involves many details, the toolbox is designed for easy use. It can be used to analyze a single trajectory estimate, as well as compare different algorithms on many datasets (e.g., this paper) with one command. The user only needs to provide the groundtruths and estimates of desired format and specify the trajectory alignment method. The toolbox generates (almost) paper-ready plots and tables. In addition, the evaluation can be easily customized.

Evo is a Python evaluation tool for SLAM systems that is simple to use [**?**]. It can evaluate SLAM system measurements and output estimates. Evo can support many kind of data sets, such as the TUM trajectory file, KITTI pose and ROS bag etc. The evo_traj euroc data.csv function can output the pose data directly, and also can draw the trajectory xyz_view rpy_view trajectory over time. And the function called evo_ape, which means absolute pose error, often used as an absolute trajectory error. The attitude relation is used to directly estimate and reference the corresponding attitude. Then, the statistics for the entire trajectory are calculated. This is useful for testing global consistency of trajectories. The relative attitude error is compared to the motion in direct comparison to the absolute attitude (attitude triangle). The measurement gives a view of the local accuracy, namely drift.

5 How to Reproduce the Project

5.1 Environment

C++ with opencv(To achieve the formulas), MATLAB 2016b(To display the final result)

5.2 How to run the whole programme

cd code/v3/
revise the variable 'cv::String pattern' in the main function of factorization.cpp to be your data path
g++ factorization.cpp -o factorization
./factorization
run the construct.m in matlab

5.3 How display the result

cd code/v3/
run earth_display.m in matlab

6 Conclusion

In conclusion, we first tried to construct the matrix W for the rest of the work. However, bacause of the size of the dataset, our computer's memory are to small to store the feature points' coordinate information. So we had to batch the dataset to calculate the sub matrix, then use formulas to merge the matrix W. Then the SVD decomposition was used to devide matrix W into rotation matrix and shape matrix. Finally we mapping the shape matrix onto the sphere. As shown in the result model, we can approximatelly recognize several continents. But some of then are still abstract, we will make some update in the later research.