SLAM-Zero

Garrett A. Partenza

March 24, 2022

Abstract

SLAM-Zero is a toy implementation of visual-based monocular simultaneous localization and mapping, where an agent builds a three-dimensional map of its environment while moving through it. This project represents an effort to grasp the fundamental mathematical foundations behind epipolar geometry while gaining experience with open-cv in python.

1 Introduction

SLAM stands for simultaneous localization and mapping, where an agent attempts to construct or update a map of an unexplored environment whilst simultaneously keeping track of the agent's location within it. The inspiration for this project came after watching George Hotz's eight-hour long Twitch stream where he developed a similar implementation. Major resources consulted during the development process notably include a visual slam textbook open source on GitHub, both original ORB-SLAM papers, and the Stanford epipolar geometry notes [1] [2] [3] [4]. The remainder of this article discusses fundamental mathematics of visual-based monocular SLAM, relevant files within the code repository, as well as weaknesses and future improvements.

2 Methodology

The algorithm in my approach follows that of a three step approach. First, given a pair of images taken sequentially of the environment, generate feature matches. Second, use the feature matches to calculate the pose transformation, notably the rotation and translation of the camera. Finally, use the pose transformation to triangulate the relevant key points. These three steps are described in greater detail in the following paragraphs.

2.1 Feature Matching

Feature matching is the process of detecting key pixels in two images and using descriptors to find corresponding matches. Features are pairs of key points and descriptors, where key points are the two dimensional image coordinates and descriptors are a feature vector which attempts to describe the key point in n-dimensional space. Key points are detected using good features to track, where key points are assumed to be the prominent corners in an image [5]. Next, orb features are computed using rotated BRIEF algorithm as in [6]. Matches between key points within both images are derived by brute force with the KNN algorithm, where the top two matches are retained. The ratio test is applied to the top two matches in order increase quality as described in [7].

2.2 Pose Transformation

Pose transformation is the process of using matches between two images to find the camera rotation and translation. To do this, the fundamental matrix is calculated using the eight-point algorithm, whereby its derivation can be reduced to solving a system of linear equations given eight or more matches. This process can be described in the following equalities, where F is the fundemental matrix and x is a set of corresponding key points.

$$x^T F x = 0 \tag{1}$$

$$\mathbf{F} = \begin{bmatrix} f_{11} & f_{12} & f_{13} \\ f_{21} & f_{22} & f_{23} \\ f_{31} & f_{32} & f_{33} \end{bmatrix} \tag{2}$$

$$uu'f_{11} + vu'f_{12} + u'f_{13} + uv'f_{21} + vv'f_{22} + v'f_{23} + uf_{31} + vf_{32} + f_{33} = 0 \tag{3}$$

Next, the intrinsic parameter matrix is used in combination with the fundamental matrix to calculate the essential matrix.

$$K^T \cdot F \cdot K \tag{4}$$

Finally, the rotation matrix R and translation vector t are calculated by decomposing the essential matrix and conducting a clarity check to ensure all points have positive depth. Together R and t create a tuple that performs a change of basis from the first camera's coordinate system to the second camera's coordinate system.

2.3 Triangulation

Triangulation in visual-based monocular slam involves using R, t, to find the three dimensional world point of a key point matched between two images. To do this, the projection matrix is created by horizontally stacking R and t, before using least squares to cast the image point into three dimensional space as in [8]. Finally, in order to improve the mapping quality, the reprojection error of each world point is calculated by reprojecting the point back onto the image frame. Points that create large reprojection errors are masked.

3 Relevant Files

1. slam.py
   (a) reads video and executes slam
   (b) plots map
2. extract.py
   (a) calculate key points and descriptors
   (b) retain previous frame key points and descriptors
3. pose.py
   (a) calculate R and t
4. triangle.py
   (a) triangulate world points

4 Results

Considering the complexities of epipolar geometry and the limited time I had to implemented the project, I am overall satisfied with my algorithms performance. One of the weaknesses of my implementation is that it does not include a back end optimization such as in the original ORB-SLAM paper. As a result, if world points are continuously plotted without reseting the axis after each frame, pose transformation error grows linearly and results in a erroneous mapping. Given more time, this would be my focus for improvement.

References

[1] Xiang Gao, Tao Zhang, Yi Liu, and Qinrui Yan. 14 lectures on visual slam: From theory to practice. *Publishing House of Electronics Industry*, 2017.

[2] Raul Mur-Artal, J. M. M. Montiel, and Juan D. Tardos. Orb-slam: A versatile and accurate monocular slam system. *IEEE Transactions on Robotics*, 31(5):1147–1163, Oct 2015.

[3] Raul Mur-Artal and Juan D. Tardos. Orb-slam2: An open-source slam system for monocular, stereo, and rgb-d cameras. *IEEE Transactions on Robotics*, 33(5):1255–1262, Oct 2017.

[4] Kenji Hata and Silvio Savarese. Computer science 231 a course notes 3: Epipolar geometry. 2011.

[5] Jianbo Shi and Carlo Tomasi. Good features to track. pages 593–600, 1994.

[6] Michael Calonder, Vincent Lepetit, Christoph Strecha, and Pascal Fua. Brief: Binary robust independent elementary features. In Kostas Daniilidis, Petros Maragos, and Nikos Paragios, editors, *Computer Vision – ECCV 2010*, pages 778–792, Berlin, Heidelberg, 2010. Springer Berlin Heidelberg.

[7] David G. Lowe. Distinctive image features from scale-invariant keypoints. *Int. J. Comput. Vision*, 60(2):91–110, November 2004.

[8] R. I. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, ISBN: 0521540518, second edition, 2004.