A Semi-Dense Feature-based VSLAM System

Zhen Li^{a,*}, Yuren Du^a, Lifeng Zhang^b

^a Yangzhou University, Yangzhou, 225000, China

^b Kyushu Institute of Technology, Kitakyushu, 804-0015, Japan

* Corresponding Author: li.zhen799@mail.kyutech.jp

Abstract

SLAM⁽¹⁾ (Simultaneous Localization and Mapping) technology is the basis of a mobile robot, also one of the most important conditions for robot intelligence. VSLAM (Visual Simultaneous Localization and Mapping) technology uses visual sensors to realize the function that a moving robot builds incremental environment maps and simultaneously realizes the self-positioning. Currently, feature-based SLAM approaches are to build sparse maps, but we can't realize robot path navigation and obstacle avoidance by using sparse maps. Therefore, this paper proposed a semi-dense VSLAM with monocular cameras. Our semi-dense mapping was optimised by local bundle adjustment and loop closing, and the depth measurement of the images used the ways of polar search and block matching. We evaluated our system in our laboratory, and it performed well and steadily.

keywords: VSLAM, Monocular, Semi-dense, Feature-based.

1. Introduction

Simultaneous positioning and map construction algorithms are used to track the camera's current position while creating an environment map based on sensor datas. VSLAM is a solution that uses visual sensor to perceive the environment around the camera. Real-time, high-accuracy and robust SLAM algorithm are all fundamental condition for a robot to realize the components of navigation, path planning and autonomous navigation. Currently, SLAM technology is mostly applied in the fields of self-driving cars, sweeping robots and unmanned aircraft. Compared with traditional laser sensor, high-precision IMU and GPS sensors, the camera costs less and consumes less power. With the improvement of algorithm and processor performance, the accuracy of VSLAM is also improving. When we come to the areas where GPS signals are not available, such as enclosed rooms and remote

mountains, VSLAM is irreplaceable.

At present, the main research direction of VSLAM is feature-based sparse SLAM and direct dense VSLAM. ORB-SLAM⁽⁹⁾ and LSD-SLAM are two typical methods. ORB-SLAM algorithm can complete real-time tracking and mapping regardless of indoor or outdoor environment, and execute closed-loop detection and relocation very well. LSD-SLAM is a direct monocular SLAM algorithm that can construct large-scale environment maps, estimate camera pose accurately based on direct image registration, and reconstruct semi-dense depth image by key frame pose. However, ORB-SLAM is more robust in scenarios where illumination changes obviously.

The sparse maps constructed by feature-based⁽⁶⁾ VSLAM system perform badly to realize the robot navigation and obstacle avoidance. Based on the feature-based monocular ORB-SLAM system, this paper proposes a feature-based semi-dense map reconstruction technique.

2. The outline of Scheme Design

2.1 System master plan

This system is composed of the following four threads: tracking⁽⁴⁾, local mapping, loop closing, semi-dense mapping, the system structure diagram is shown in Fig.1.

2.2 Tracking

The main idea is to find more congruent relationship between the current frame and the local map, in order to optimize the pose of current frame. The following three motion models in the Tracking are all used to get the pose of the current frame.

(a) Tracking with Motion Model

Assuming that the object was moving at a uniform speed, the camera pose of the current frame could be estimated by the pose and velocity of the last frame. The speed

Fig. 1. System structure diagram

of the last frame can be calculated by several previous frames. This model is suitable for the cases that camera is moving along with the straight line at a uniform speed.

(b) Tracking with Reference key frame

If the motion model has failed, you can try to match the last key frame, because the current frame is not far from the last key frame. We can use BoW(Bag of Words) to speed up the match process. Firstly, we calculate the BoW of the current frame and set the initial position as the pose of the last frame; Secondly, we need to search for feature matching according to pose and BoW dictionary; Finally, the matching feature is used to optimize camera pose.

(c) Relocalization

If the current frame also fails to match the nearest neighbor key frame, this means that the current frame has been lost. At this point, you have to match all the key frames to find the right place. Firstly, calculate the BoW vector of the current frame. Secondly, several key frames are chosen as alternatives by BoW dictionary. Thirdly, key frames with enough feature points are selected. Finally, the camera pose can be determined by feature point matching. We tend to select the key frame that has enough interior feature points.

2.3 Local Mapping

The current key frame is inserted into the map, but some key frames are found redundant after judgement, we have to delete it. Due to triangulation error, originally one point may be predicted into two points by mistake. The two points need to be fused into one.

The local map is associated with the previous tracking and subsequent loop closing, and the key frame is generated by the tracking thread, while the condition of local bundle adjustment for the local map is that there is no loop and the key frame queue is empty. The elements contained in the map are useful for loading and saving maps.

2.4 Loop Closing

Cyclic detection mainly consists of two steps: 1. Cyclic information detection and confirmation; 2. Modify and optimize the pose diagram. Regardless of monocular, binocular or RGBD camera, all of them exist the problem of measurement drift. As the path continues extending, the error of the previous frames will be transmitted to the following frames, resulting in a very large error of the pose of the last frame in the world coordinate system. In addition to using optimization methods to adjust the local and global pose, loop closing can also a good way to optimize the pose.

In other words, when the camera is moving, it returns to the same pose as before, the map forms a closed loop. The camera can automatically identify the previous location, and then transmit this information back to the entire map to get

a fairly accurate map information, this is what loop closing works. Therefore, loop closing is a very useful method for large-scale⁽³⁾ map reconstruction.⁽⁷⁾

2.5 Semi-dense Mapping

Suppose there is a sequence of video, and we get the trajectory of each frame. Now we take the first image as the reference frame and calculate the depth of each pixel in the reference frame. First, we should find out how we completed the process in the feature-based section:

1. First, we extract feature points from images and calculate the matching between feature points according to descriptors. In other words, by means of feature points, we track a point in space and know its pose in different images.

2. Then, since we cannot determine the pose of feature points with only one image, we must estimate its depth through observation from different perspectives, we will use the triangulation here.

In the semi-dense depth map estimation, the difference is that we cannot compute descriptors with every pixel as an feature point. So matching becomes an important part in the semi-dense⁽⁸⁾ depth estimation problem. Now we requires the use of polar search and block matching techniques. When we know the position of pixels in each graph, we can use triangulation, like in the feature-based SLAM, to determine its depth. However the difference is that we're going to use triangulation for many times to make the depth estimation converge, not just one time. We hope that the depth estimation can gradually converge from an uncertain number to a stable number as the measurement times increasing, this is the depth filter technique⁽²⁾.

3. The main algorithm of the system

3.1 Polar search and block matching

As is shown in Fig.2. We can see the geometry of same point from different angles. The camera on the left detects a certain pixel p_1 . Since this is a monocular camera, we cannot measure its depth, so its depth could be in some fixed region, instead the number is between the minimum and infinity. Therefore, the corresponding spatial points of the pixel are distributed on a line segment. From another perspective, the projection of this line segment also forms a line on the image plane, which is known as a polar line. When we know the motion between the two cameras, this polar line can also be determined.

In the feature-based method, we find the position of p_2

Fig. 2. Polar search diagram

through feature point matching. However, we do not have descriptors, so we can only search the polar line for points that are similar to p_1 . Specifically, we might go from one end of the polar line to the other of the second image, one by one, comparing similarity of each pixel to p_1 . From the view of direct pixels comparison, this approach is similar to the direct method.

In the direct method, we also know that comparing the brightness value of a single pixel is not very stable and reliable. Since the brightness of individual pixels is indistinguishable, we can try to compare pixel blocks. We take a small piece of size $w \times w$ around p_1 , and then take many small pieces of the same size on the polar line for comparison, to some extent we improved the discrimination, this is block matching. During this process, the comparison only makes sense when we assume that the entire chunk grayscale remained invariability across different images.

Now we take the little pieces around p_1 , and we take a bunch of little pieces on the polar line. We might call the small pieces around p_1 as $A \in R^{w \times w}$, and call the small pieces on the polar line as $B_i, i = 1 \dots n$. To calculate the difference between small blocks, we can take the NCC method:

NCC (Normalized Cross Correlation): This approach is a bit more complex, calculating the correlation between two small pieces:

$$S(A, B)_{NCC} = \frac{\sum_{i,j} A(i, j)B(i, j)}{\sqrt{\sum_{i,j} A(i, j)^2 \sum_{i,j} B(i, j)^2}} \quad (1)$$

The correlation that is close to zero, means two images are not similar, while a correlation that is close to one means very similar. There is a contradiction between precision and

efficiency in this calculation method. Methods with good accuracy⁽⁶⁾ often requires complex calculations, while simple and fast algorithms are often ineffective. This requires a balance in practical engineering.

3.2 Gauss depth filter

The pixel depth estimation can also be converted to a state estimation problem. There are two solutions: filter and non-linear optimization. Although the nonlinear optimization affection is better, considering that the front end has occupied a lot of computational load in the case of SLAM, which has a strong real-time⁽⁵⁾ requirement, the filtering method with less computational load is more suitable for the map construction, so we use Gauss Distribution here.

Set the depth of a pixel to obey:

$$P(d) = N(\mu, \sigma^2) \quad (2)$$

Every time a new data comes in, we will calculate its depth. Similarly, suppose that this observation is also a Gauss Distribution:

$$P(d_{obs}) = N(\mu_{obs}, \sigma_{obs}^2) \quad (3)$$

How to use the observed information to update the original distribution of d , is exactly an information fusion problem. According to appendix A, we understand that the product of two Gauss Distributions is still a Gauss Distribution. Assuming that the distribution of fused d is $N(\mu_{fuse}, \sigma_{fuse}^2)$, then according to the product of Gauss Distribution, we can get:

$$\mu_{fuse} = \frac{\sigma_{obs}^2 \mu + \sigma^2 \mu_{obs}}{\sigma^2 + \sigma_{obs}^2} \quad (4)$$

$$\sigma_{fuse}^2 = \frac{\sigma^2 \sigma_{obs}^2}{\sigma^2 + \sigma_{obs}^2} \quad (5)$$

3.3 Uncertainty analysis

Considering a certain polar line search, we found the corresponding point p_2 of p_1 , and observed the depth value of p_1 , which is shown in fig.3. We think that the 3D point corresponding to p_1 is P . Therefore, we can call O_1P as p , O_1O_2 as the translation t of the camera, and O_2P as a . And I'll call the bottom two angles of this triangle α , β . Now, considering that there is a pixel size error on the polar line l_2 that makes the β Angle become β' , and p becomes p' , and call the upper angle γ . So we need to figure out how much the error of this one pixel cause from p' to p .

Fig. 3. Uncertainty analysis

This is a typical geometry problem. So we write the geometry of this quantity:

$$a = p - t \quad (6)$$

$$\alpha = \arccos \langle p, t \rangle \quad (7)$$

$$\beta = \arccos \langle a, -t \rangle \quad (8)$$

Disturbance of p_2 by one pixel will produce a delta beta change, since the camera focal length is f :

$$\delta\beta = \arctan \frac{1}{f} \quad (9)$$

$$\beta' = \beta + \delta\beta \quad (10)$$

$$\gamma = \pi - \alpha - \beta' \quad (11)$$

$$\|p'\| = \|t\| \frac{\sin \beta'}{\sin \gamma} \quad (12)$$

Thus, we determine the depth uncertainty caused by the uncertainty of a single pixel. If it is considered that the block matching of polar line search has only one pixel error, then it can be set as follows:

$$\sigma_{obs} = \|p\| - \|p'\| \quad (13)$$

In practical engineering, when the uncertainty is less than a certain threshold, the depth data can be considered to have converged. In summary, we present a complete process for estimating dense depth:

- (a) Assume that the depth of all pixels satisfies some initial Gauss Distribution.
- (b) When a new data is generated, the pose of projection point is determined through polar line search and block matching.

Table 1. Experiment environment

OS	Ubuntu16.04
PROCESSOR	TEGRA X1
ROS	Kinetic
LANGUAGE	C++

Fig. 4. Semi-dense VSLAM of a view in the lab

- (c) Calculate the depth and uncertainty after triangulation according to geometric relations.
- (d) Integrate current observations into previous estimates. Stop calculation if convergence, otherwise return step(b).

These 4 steps constitute a set of feasible depth estimation methods.

4. Simulation

4.1 Experiment environment

We test the project in our laboratory, and used the intelligence sweeper to run the SLAM system, the experiment environment is shown in the Table 1.

4.2 Results of the small-scale environment

The picture is a view of my laboratory, and we can reconstruct the semi-dense map of the picture, the point cloud in the picture is more dense, and the feature points is much more too, which is shown in the Fig.4.

4.3 Results of big-scale environment

We walked for a round around the desk with the camera, found that the approach of this thesis is very robust and finish the task very well. And when we go back to the initial position, this approach can detect a loop and optimize the whole map. As is shown in Fig.5.

Fig. 5. Loop closing test

5. Conclusions

The proposed semi-dense monocular VSLAM based on ORB features is more robust and efficient than the direct VSLAM method. Due to the existence of loop closing, the target can be quickly recovered after losing very easily. Semi-dense VSLAM is important for semantic segmentation, object recognition and 3D environment reconstruction⁽¹⁰⁾. So the semi-dense VSLAM will make a great contribution to future mobile robot development. This technology makes it become possible for mobile robots to possess the components of navigation, path planning and autonomous navigation.

Acknowledgment

This work was supported by Yangzhou University International Academic Exchange Fund.

References

- (1) R. Mur-Artal, J. Montiel, J. Tardos: "ORB-SLAM: A versatile and accurate monocular SLAM system", IEEE Trans. Robot, vol. 31, no. 5, pp. 1147-1163, 2015.
- (2) B.D. Gouveia, D. Portugal, L. Marques: "Speeding up Rao-Blackwellized particle filter SLAM with a multithreaded architecture", Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp. 1583-1588, 14-18, 2014.
- (3) H. Strasdat, J. Montiel, A.J. Davison: "Scale drift-aware large scale monocular SLAM", Proceedings of the Robotics: Science and Systems (RSS), pp. 27-30, 2010.
- (4) Berta Bescos, José M. Fàcil, Javier Civera, José Neira: "DynaSLAM: Tracking, Mapping and Inpainting in Dynamic Scenes", IEEE Robotics and Automation Letters. Volume: 3. Issue: 4, 2018
- (5) Jiexiong Tang, Ludvig Ericson, John Folkesson, Patric Jensfelt: "Efficient Correspondence Prediction for Real-Time SLAM", Computer Vision and Pattern Recognition, 2019.

- (6) Raul Mur-Artal, Juan Domingo Tardos: "Probabilistic Semi-Dense Mapping from Highly Accurate Feature-Based Monocular SLAM", *Robotics: Science and Systems*, 2015.
- (7) A. Geiger, J. Ziegler, and C. Stiller: "Stereoscan: dense 3d reconstruction in real-time", *Intelligent Vehicles Symposium (IV)*, IEEE, pp. 963-968, 2014.
- (8) Engel, Sturm, Cremers: "Semi-dense visual odometry for a monocular camera", *International Conference on Computer Vision (ICCV)*, 2013.
- (9) E. Rublee, V. Rabaud, K. Konolige, and G. Bradski: "ORB: an efficient alternative to SIFT or SURF", in *Proc. of the IEEE Intl. Conf. on Computer Vision (ICCV)*, vol. 13, 2011.
- (10) Hornung A, Wurm K M, Bennewitz M, et al. OctoMap: "An efficient probabilistic 3D mapping framework based on octrees", *Autonomous Robots*, 34(3): 189-206, 2013.