Journal of Computers Vol. 28, No. 5, 2017, pp. 316-326
doi:10.3966/199115992017102805027

An Application Oriented Self-adaptive SLAM System for Mobile Robots

Lin-Jun Li[12*], Yi-An Zhou[13]

[1] Beijing Engineering Research Center of Smart Mechanical Innovation Design Service, Beijing, China

[2] School of Astronautics, Harbin Institute of Technology, Harbin, China
1140410201@hit.edu.cn

[3] School of Information & Communication Engineering, Bejing Information Science & Technology University, Beijing, China
16550942@qq.com

Received 24 July 2017; Revised 24 August 2017; Accepted 24 September 2017

Abstract. Since more and more applications such as automatic drive and tasks for quadcopters rely on real-time Simultaneous Localization and Mapping system, it's necessary to adjust the algorithm to adapt to real time tasks. Based on feature based methods like ORB SLAM, we try to propose an approach to further enhance efficiency by speed self-adaptive key frame and key feature point selection, which contributes to decrease workload on hardware system, simplifies the procedure of parameter calibration and improve robustness of the whole system.

Keywords: feature based SLAM, machine vision, optimization, robotics

1 Introduction

In the field of VSLAM (Visual Simultaneous Localization and Mapping), feature based SLAM has become been one of the most popular directions since the most widely known method ORB-SLAM [1] was proposed and improved for years. In contradict to dense SLAM, or direct SLAM [2], which works on continuous time and optimize the trajectory through dense, direct image alignment, depending on powerful computing capacity of GPUs, ORB-SLAM uses a different feature detecting method with key frame selection and proposed a system structure with three parallel threads, which is tacking, local mapping and loop closing, which contributes to improve efficiency to work as a real-time system like automatic driving.

However, ORB-SLAM system still depends on manual adjustment for specific robotic system to adapt to move speed and camera location, and not effect when camera is rotating or moving in a high speed. Besides, although there has been some attempt to reduce error using IMU(Inertial measurement unit) data with data-fusion methods [3], most of them are based on KF (Kalman Filter) or EKF (Extended Kalman Filter), which rely on the linear assumption and thus couldn't work very well in different complex environments.

Based on Mono-ORB-SLAM, we focus on tracking as well as local mapping and try to reduce error with data-fusion in a different way, which is adjust feature point sample size by expectation of velocity from IMU and Visual Odometry [4], in order to improve the efficiency and robustness of VSLAM system for real-time application.

About other SLAM approaches, RGB-D sensors provide high-precision color and depth information at the same time, and stereo methods improve the accuracy, both of which require extra computational

* Corresponding Author

capacity to promote the efficiency.

Although there are other CNN based back-end SLAM systems [5], those models with high computational complexity are unlikely to be used on embedded systems or processors for industrial robots and unnecessary for parameter identification.

Since in most situation, the point cloud created by feature-based VSLAM system, for instance, ORB SLAM, is usually too sparse to build a precise map, we are trying to explore some new application of VSLAM method like visual servo and domestic robot, which also depends on other technology including background reconstruction and object detection or tracking, and may related to some foundation of machine learning.

2 Related Work

You may prepare your camera-ready manuscript with MS-Word using this typeset together with the template joc.dot (see Sect. 3) or any other text processing system. In the latter case, please follow these instructions closely in order to make the volume look as uniform as possible.

We would like to stress that the class/style files and the template should not be manipulated and that the guidelines regarding font sizes and format should be adhered to. This is to ensure that the end product is as homogeneous as possible.

2.1 Camera Model

Most application of computer vision are based on pinhole camera model, in which the relationship between coordinates of every point in 3D space could be described by their projection on image plane.

In this model, the relationship between the coordinate of the same point in real world and on the picture is:

$$s\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_x \\ 0 & 0 & 1 \end{bmatrix}\begin{bmatrix} x \\ y \\ z \end{bmatrix} \tag{2-1}$$

In the equation, s is ratio factor, and the matrix about f and c is the internal parameter of the camera, which could not be revised after the camera being produced.

Such relationship also exist for RGB-D camera:

$$\begin{bmatrix} x \\ y \\ z1 \end{bmatrix} = \begin{bmatrix} \dfrac{(u-c_x)z}{f_x} \\ \dfrac{(v-c_y)z}{f_y} \\ z = \dfrac{dep(u,v)}{s} \end{bmatrix} \tag{2-2}$$

In most situation, inevitably there are distortion because the lens are not perfect spherical surfaces. Radial distortion could be corrected with software through a series of algorithms [6], and a realistic camera model usually has the following form:

$$\begin{bmatrix} x_u \\ y_u \end{bmatrix} = \begin{bmatrix} x_d + (x_d - x_c)(K_1 r^2 + K_2 r^4 + \cdots) + \cdots \\ y_d + (y_d - y_c)(K_1 r^2 + K_2 r^4 + \cdots) + \cdots \end{bmatrix} \tag{2-3}$$

And the calibration could be completed easily with MATLAB or OpenCV tuning tools.

With a reasonable camera model, the motion of the camera or the mobile robot could be described as:

$$\begin{bmatrix} u \\ v \end{bmatrix} = C + (Rp + t) \tag{2-4}$$

And:

$$\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \sim \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & r_{1a} & t_1 \\ r_{21} & r_{22} & r_{2a} & t_2 \\ r_{a1} & r_{a2} & r_{a3} & t_3 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \quad \textbf{(2-5)}$$

Which is also known as PnP problem, which could be solved with algorithms provided by OpenCV (like SolvepnpRANSAC function) or PCL (ICP function).

2.2 Key Frame

ORB SLAM proves an alternative framework for real time SLAM system inherit from PTAM [7], which lacks a better key frame selection method and automatic initialize method.
ORB SLAM has three main threads of tracking, local mapping and loop closing. In tracking method, one of the most important component is the key frame selection module. Fig. 1 if the flow chart of tracking thread.

In ORB SLAM, besides monocular SLAM, RGB-D [8] and stereo SLAM are also build on monocular ORB(Oriented FAST and Rotated BRIEF) feature [9], which is known to be much faster than SIFT and widely used in object detection, which pose estimation and tracking local map are based on. Camera pose could be traced with different motion model, then key frames generating depends on relativity of local maps. The score of each key frame could be calculated by max common words, by which candidates of key frames would be grouped together and irrelevant key frames will be eliminated.

Fig. 2 shows the different relationship between key frames between three models provided for key frame selection. Tracking with model is used for uniform motion cameras, while tracking with reference key frame, in which bag of words method is used to accelerate the match process, and relocalization are used for other random motion, which usually used when the relativity between key frames is undesirable.

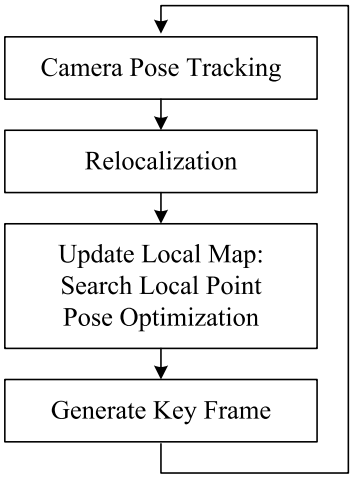


Fig. 1. Procedure of tracking loop

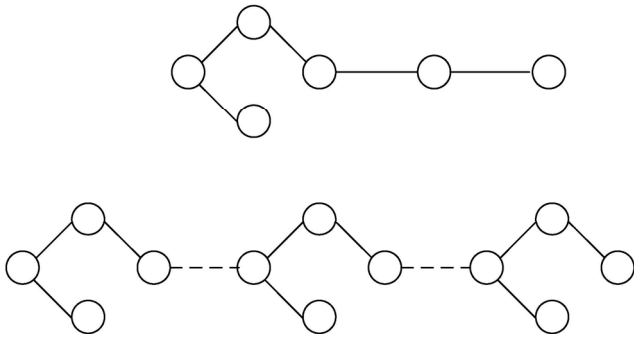


Fig. 2. Key frame selection

2.3 Pose Estimation

Based on projection theory, motion only bundle adjustment optimizes pose of the camera [10], which could be represented as: for $R \in SO(3)$:

$$r'_{r,i} = sR(r'_{r,i}) - r'_0 \quad \textbf{(2-6)}$$

And:

$$r'_0 = r_0 - r_r + sR(r_r) \quad \textbf{(2-7)}$$

We translate it in to an optimize model:

$$\sum_{i=1}^{n}\left\|e_i\right\|^2=\sum_{i=1}^{n}\left\|r_{r,i}^{'}-[sR(r_{r,i}^{'})+r_0^{'}]\right\|^2$$

$$=\sum_{i=1}^{n}\left\|r_{r,i}^{'}-[sR(r_{r,i}^{'})+r_0^{'}]\right\|-2r_0^{'}\sum_{i=1}^{n}\left\|r_{r,i}^{'}-sR(r_{r,i}^{'})\right\|+n\left\|r_0^{'}\right\|^2 \quad \textbf{(2-8)}$$

The optimize model is irrelevant to r_0:

$$\{R,t\}=\arg\min\sum_{i=1}^{n}\left\|e_i\right\|^2 \quad \textbf{(2-9)}$$

In order to translate 2D point to 3D point, for matched 2D key point x_1 and x_2, transform T_i could convert x_i to 3D point:

$$T_2X_2=R(T_1X_1)+t \quad \textbf{(2-10)}$$

And according to machine vision assumptions and camera model, views should be in same plane:

$$N^TX_1=d \quad \textbf{(2-11)}$$

Subtitution:

$$T_2X_2=R(T_1X_1)+t\frac{1}{d}N^T(T_1X_1) \quad \textbf{(2-12)}$$

Thus:

$$X_2=T_X^{-1}\left(R+t\frac{1}{d}N^T\right)(T_1X_1) \quad \textbf{(2-13)}$$

In conclusion, the tracking thread basically works with front end image processing and provide data for local mapping, including a visual odometry which is very important for robotic control system.

2.4 Graph Optimization and Mapping

SLAM problems could be described as many different representations, including the traditional EKF represents, non-linear optimization or bundle adjustment [11].

Graph Optimization is also known as global SLAM, which represent the motion constraint and variables as a Graph over time showed in Fig. 3:

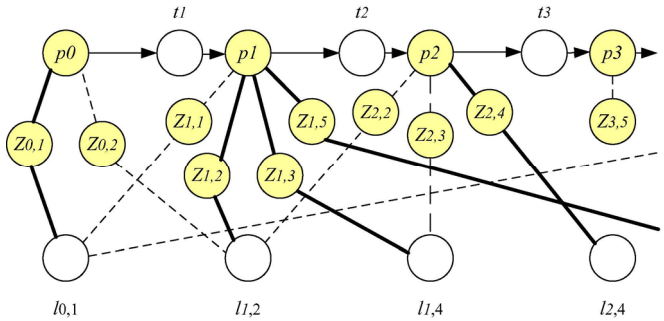


Fig. 3. Graph optimization

$$G=\{V,E\} \quad \textbf{(2-14)}$$

V represents motion constraints as sides and E represents variables as nodes, thus as time passes if the error is minimized, we get our optimization solution.

Assuming the error are normally distributed:

$$e_{v,k} = x_k - f(x_{k-1}, h_k) \tag{2-15}$$

$$e_{y,k,j} = z_{k,j} - g(x_k, y_j) \tag{2-16}$$

The objective function could be described as:

$$J(x, y) = \sum_{k=1}^{N} e_{v,k}^T \Sigma_{v,k}^{-1} e_{v,k} + \sum_{k=1}^{N} \sum_{j=1}^{M} e_{y,k,j}^T \Sigma_{y,k,j}^{-1} e_{y,k,j} \tag{2-17}$$

In which the former term is model error and the later model is observation error.
Linear approximation:

$$e(x + \Delta x) = e(x) + J\Delta x \tag{2-18}$$

Thus:

$$E(x + \Delta x) = e^T(x + \Delta x)\Sigma^{-1} e(x + \Delta x) = ((e + J\Delta x)^T)\Sigma^{-1}(eJ\Delta x)$$

$$= E(x) + 2e^T \Sigma^T J\Delta x + \Delta x^T \Sigma^{-1} J\Delta x \tag{2-19}$$

To optimize ΔE ,

$$\frac{\partial \Delta E}{\partial \Delta x} = 0 \tag{2-20}$$

$$2J^T \Sigma^{-1} J\Delta x + 2J^{-1}\Sigma^{-1} e = 0 \tag{2-21}$$

Solution:

$$J^T \Sigma^{-1} J\Delta x = -J^{-1}\Sigma^{-1} e \tag{2-22}$$

The equation is linear in which J could be calculated with e. The problem could be solved by Gauss-Newton iteration or some other gradient search method.

3 Self-Adaptive Improvement

3.1 Tracking

In original ORB SLAM algorithm, the threshold value is calculated with magic numbers:

$$\min \text{Commons} = 0.8 \times \max \text{Common Words} \tag{3-1}$$

$$\min \text{Score To Retain} = 0.75 \times \text{best Acc Score} \tag{3-2}$$

Although such compute method are designed for uniform motion state, to improve the robustness and self-adaptable, it's proper to assume the ideal threshold follows linear model with constant parameters, and the parameter could be determined by a series of experiments:

$$\text{Threshold} = (C - C_v v) \times \max \text{Common Words} \tag{3-3}$$

For robots or vehicles moving in a high speed, the frame rate should increase to maintain the precision in case losing track, while those moving slower should save their computational resource for mapping or other application.

3.2 Key Point Selection for Pose Estimation

The Idea of pose estimation relies on key points, which is filtered by ORB feature point matching. The most widely used and efficient way to filter out match points is rejecting mismatched pairs by distance constraints. [12] As the max key point number increases, the accuracy does not increase automatically, which means there would be more mismatch and cost more computation capacity. Such conclusion could

be made based on a series of experiments showed in Fig. 4.

(a) 50 points (b) 200 points

(c) 500 points (d) 1000points

Fig. 4. Matches of ORB feature

Fig. 4 indicates that, considering the definition of good matching in SLAM system, the maximum distance of key points shouldn't be larger than twice of the correct minimum distance for rotational motion which could be calculated with an average distance of a certain proportion of matched points. Fig. 5 and Fig. 6 show that when max key point number remain small, the accuracy increases while time cost doesn't change dramatically. But in larger scale, time cost is proportional to the square of max key point number.

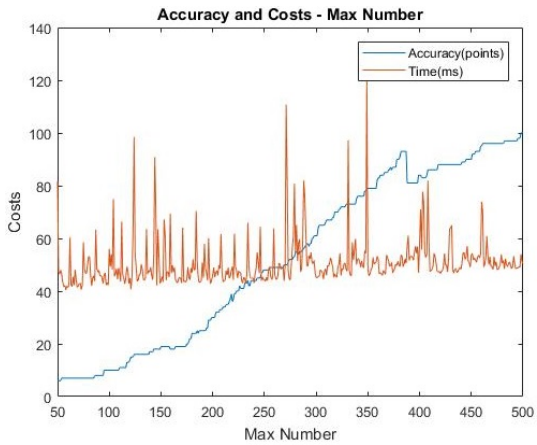


Fig. 5. Max number remain slow

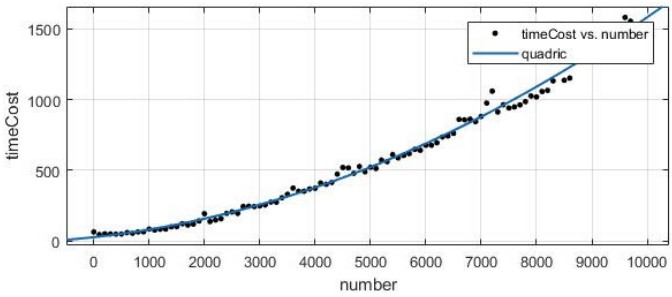


Fig. 6. Large Scale

The experiments indicate that, for real world hardware system, the most time-consuming step is feature detection instead of key point matching.

For real-time SLAM system with high robustness, the appropriate real-time key point number depends on expectation of velocity derived from the tracking thread, which also contains visual odometry.

$$N_m = N_1 - C_r \|r\| \quad \textbf{(3-4)}$$

In which N_1 is ideal upper bond depends on hardware limitation and could be identified by experiments.

3.3 Parameter Identification

Considering that when the relationship is non-linear the model may be more accurate, a statistic learning model could be applied for function approximation. Both polynomial model and linear model could be tested with criteria that certain percentage of related key frames which could still be manual adjusted, but more robust. Fig. 7 shows the basic training model of our system.

We tested the key point numbers, key frame rate and speed when the tracking thread lost track, and recorded certain proportion which in our system is half of them as proper key frame rate. Then implement PCA (Principal Components Analysis) to reduce the dimension of the model. After that we group data as training set and test set, and build regression model based on multi-layer neural network with structure showed in Fig. 8.

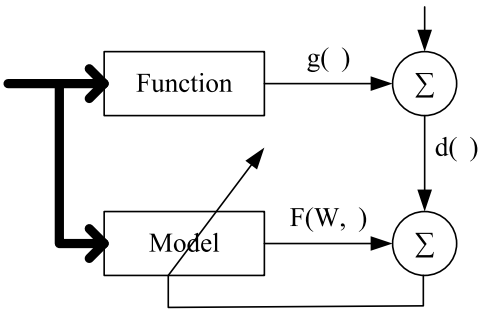


Fig. 7. Statistic learning model

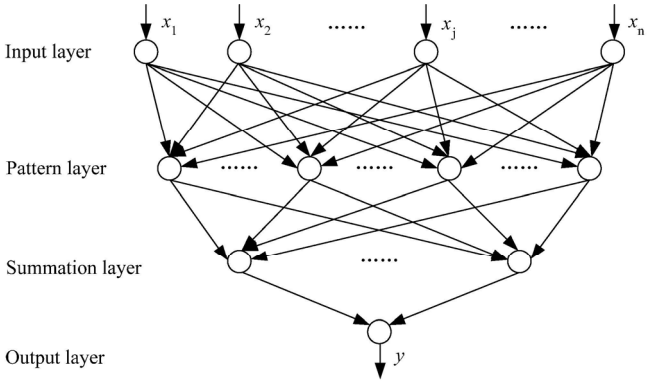
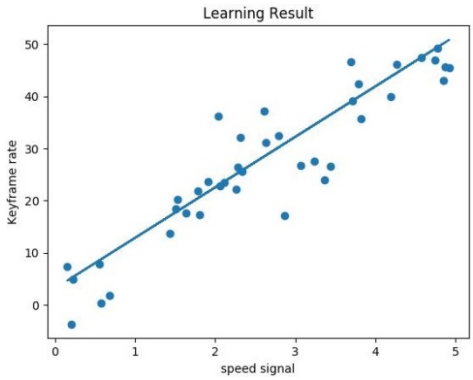
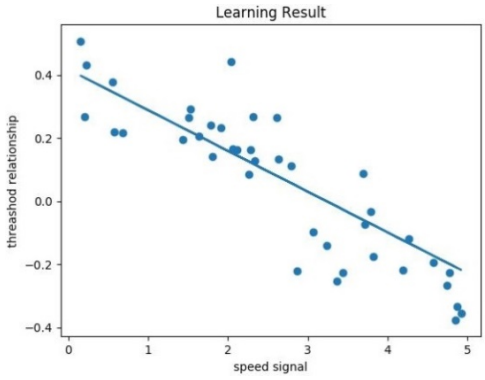


Fig. 8. Structure of regression neural network

In Fig. 9 (a) and (b), based on experiments on our ARM and Arduino based mobile robot, parameters could be calibrated with regression neural network implemented with Keras framework with Tensorflow backend.



(a)



(b)

Fig. 9. Result

Both linear model and non-linear model has been test on physical system, but the result shows that in most situation, non-linear model has heavier computing burden in real system and not necessary.

3.4 G2O Based Graph Optimization

For real time application, to ensure stability and preemptive priority scheduling, temporary increasing of tolerance of error should be acceptable. For bundle ajustmnet which could be described as graph optimize model with complexity of O(n^3) could be siplified to a O(N^2+ M N^2) problem based on the sparsity of the Jacobian.

For g2o library users, the tolerance is also an adjustable variable to improve adaptive only when mapping is not primary mission.

3.5 Motion Control and Multi-sensor Information Fusion

In order to improve the accuracy of pose estimation as well as localization, we tried to fusion other forms of information with SLAM system. [13] For modern SLAM system, lazor gyro and IMU (Inertial Measurement Unit) are common devices.

For IMU, Based on Euler angles, a linear transform could be used to derive the pose of the camera or robot [14]:

$$\begin{bmatrix}\psi\\ \theta\\ \gamma\end{bmatrix}=\begin{bmatrix}\cos\theta\sin\gamma & \cos\gamma & 0\\ -\sin\theta & 0 & 1\\ -\cos\theta\cos\gamma & \sin\gamma & 0\end{bmatrix}\begin{bmatrix}\omega_{nx}\\ \omega_{ny}\\ \omega_{nz}\end{bmatrix}=\begin{bmatrix}\dfrac{\sin\gamma}{\cos\theta} & 0 & -\dfrac{\cos\gamma}{\cos\theta}\\ \cos\gamma & 0 & \sin\gamma\\ \sin\gamma\tan\theta & 1 & -\cos\gamma\tan\theta\end{bmatrix}\begin{bmatrix}\omega_{nx}\\ \omega_{ny}\\ \omega_{nz}\end{bmatrix} \quad \textbf{(3-5)}$$

Thus we have a group of linear differential equations:

$$\begin{aligned}\psi &= \frac{\sin\gamma}{\cos\theta}\omega_{nx}-\frac{\cos\gamma}{\cos\theta}\omega_{nz}\\ \theta &= \cos\gamma\omega_{nx}+\sin\gamma\omega_{nz}\\ \gamma &= \sin\gamma\tan\theta\omega_{nx}+\omega_{ny}-\cos\gamma\tan\theta\omega_{nz}\end{aligned} \quad \textbf{(3-6)}$$

And for three-axis acceleration sensor, we have:

$$C_n=\begin{bmatrix}\cos\theta\cos\psi & \cos\theta\cos\psi & -\sin\theta\\ \sin\gamma\sin\theta\cos\psi-\cos\gamma\sin\psi & \sin\gamma\sin\theta\sin\psi+\cos\gamma\cos\psi & \sin\gamma\cos\theta\\ \cos\gamma\sin\theta\cos\psi & \cos\gamma\sin\theta\sin\psi-\sin\gamma\cos\psi & \cos\gamma\cos\theta\end{bmatrix} \quad \textbf{(3-7)}$$

Both velocity and location could be calculated with integration.

Based on minimum error method, we could derive Kalman Filter Function [14]:

$$S=\sum_{i=1}^{n}\omega_i\,(\hat{q}-q_i)^2 \quad \textbf{(3-8)}$$

The Error is minimum when:

$$\begin{aligned}\frac{\partial S}{\partial\hat{q}} &= \frac{\partial}{\partial\hat{q}}\left(\sum_{i=1}^{n}\omega_i\,(\hat{q}-q_i\right)=0\\ \sum_{i=1}^{n}\omega_i\,\hat{q}-\sum_{i=1}^{n}\omega_i\,\hat{q}_i &= 0\\ \hat{q} &= \frac{\sum_{i=1}^{n}\omega q_i}{\sum_{i=1}^{n}\omega_i}\end{aligned} \quad \textbf{(3-9)}$$

Which also indicates that we've derived a optimized estimate, and:

$$\hat{q} = \frac{\frac{1}{\sigma_1^2} q_1 + \frac{1}{\sigma_2^2} q_2}{\frac{1}{\sigma_1^2} + \frac{1}{\sigma_2^2}} = \frac{\sigma_2^2}{\sigma_1^2 + \sigma_2^2} q_1 + \frac{\sigma_1^2}{\sigma_1^2 + \sigma_2^2} q_2 \tag{3-10}$$

Thus:

$$\frac{1}{\sigma^2} = \frac{1}{\sigma_1^2} + \frac{1}{\sigma_1^2} = \frac{\sigma_1^2 \sigma_2^2}{\sigma_1^2 + \sigma_2^2} \tag{3-11}$$

In our model, q_1, q_2 are different ratio for IMU and acceleration sensor, which still need to be determined.

In order to derive the variance, we apply Bayesian formula:

$$p_1 p_2 = \frac{1}{\sigma_1 \sigma_2} e^{\frac{(q-\hat{q}_1)^2}{2\sigma_1^2} - \frac{(q-\hat{q}_2)^2}{2\sigma_2^2}} \tag{3-12}$$

So the the product is still gaussian distribution,

$$\sigma^2 = \frac{1}{\frac{1}{\sigma_1^2} + \frac{1}{\sigma_1^2}} = \frac{\sigma_1^2 \sigma_2^2}{\sigma_1^2 + \sigma_2^2} = \sigma_1^2 - \frac{\sigma_1^2}{\sigma_1^2 + \sigma_2^2} \tag{3-13}$$

And we ca optimal kalman gain and the filter equation:

$$\hat{x}_{k+1} = \hat{x}_k + \frac{\sigma_k^2}{\sigma_k^2 + \sigma_z^2} (z_{k+1} - \hat{x}_k) \tag{3-14}$$

With observation equation, we can improve the estimation of velocity and the motion control efficiency, thus enhance the self-adaptive operating in SLAM process.

4 Simulation and Implementation

Based on TUM dataset, we have simulated SLAM process in different environments. The original ORB SLAM system has great rotational errors and loses track easily as the Fig. 10 shows when time interval between two key frames become too long.

Fig. 10. Track lost

Fig. 11 indicates that in this system, SLAM process become more robust and do not lose track easily and more adaptive in different moving speed in (a) and (b). Possible reason could be higher key frame rate for high speed motion and key point selection filter out more mismatch points, which enhanced the

efficiency.

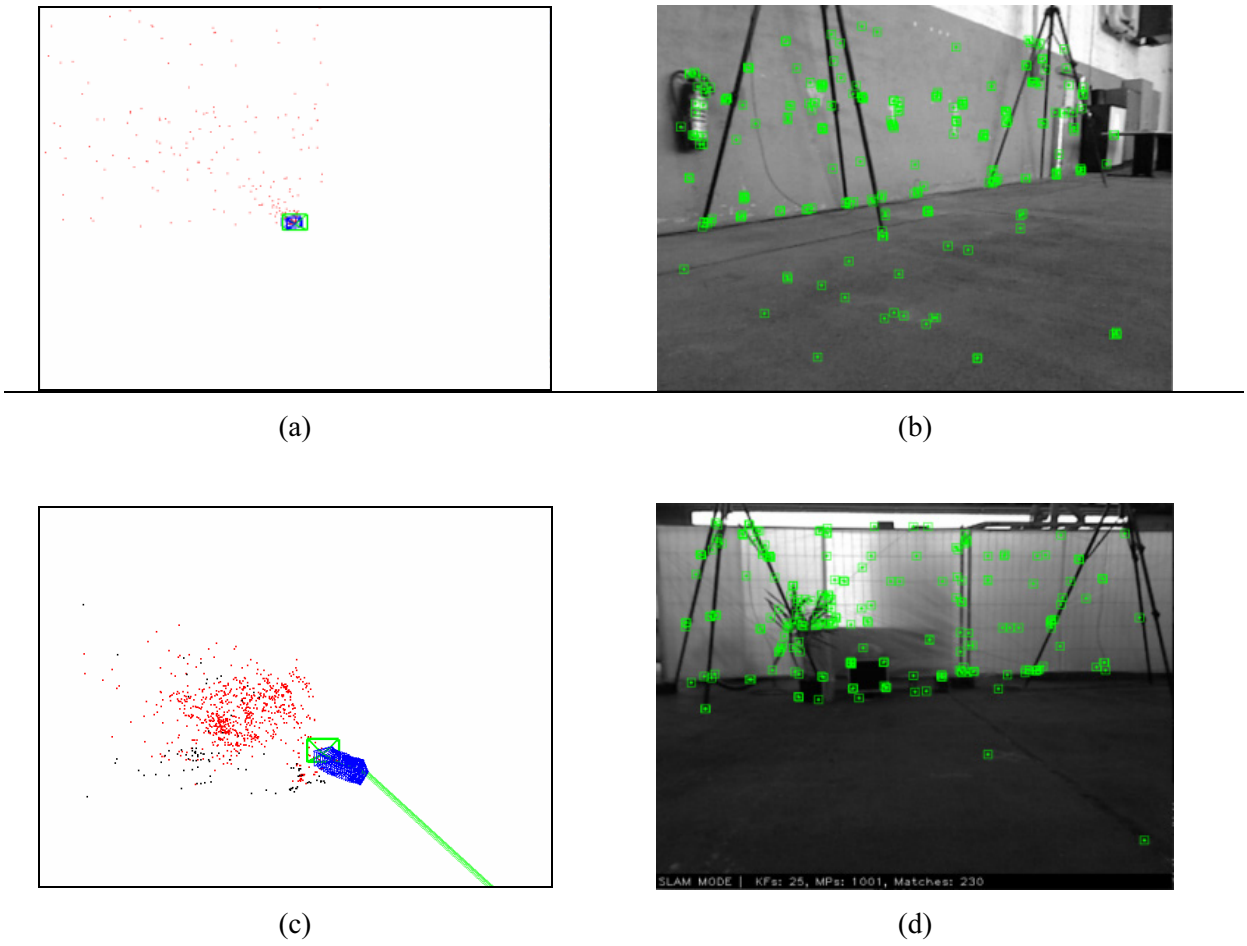


Fig. 11. Tracking and mapping

However, too much adjusting could cause instability, for instance, inappropriate parameter could lead to track lost sometimes.

5 Conclusion

Based on velocity self-adaptive key frame selection and key point selection, although map become sparser in some specific situation, the system become more robust and self-adaptive. But this system still need manual adjustment and parameter identification, which is still a problem yet to solve and could reference to some adaptive method in machine learning method.

Although convolutional neural network is not used in our SLAM system, we managed to explain using machine learning method contributes to parameter tuning and other system identification details, which could be studied further.

With the development of sensor technology and data processing, more and more sensors could be used in real-time SLAM system. In order to further improve the efficiency as well as the accuracy, it's necessary to study effective methods for multi-sensor information fusion.

With more study of feature based SLAM, there are still a lot of engineering details which could be improved in real system for real-time application. It is possible that SLAM technology combining with machine vision and machine learning is widely used in industrial field including visual servo as well as automatic drive in the near future.

References

[1] R. Mur-Artal, J.D. Tardos, ORB-SLAM2: an open-source SLAM system for monocular, stereo and RGB-D cameras, IEEE Transactions on Robotics 33(5)(2017) 1255-1262.

[2] C. Kerl, J. Stückler, D. Cremers, Dense continuous-time tracking and mapping with rolling shutter RGB-D cameras, in: Proc. 2015 IEEE International Conference on Computer Vision (ICCV), 2015.

[3] J. Civera, O.G. Grasa, A.J. Davison, J.M.M. Montiel, 1-Point RANSAC for extended Kalman filtering: Application to real-time structure from motion and visual odometry, Journal of Field Robotics 27(5)(2010) 609-631.

[4] S. Leutenegger, S. Lynen, M. Bosse, R. Siegwart, P. Furgale, Keyframe-based visual-inertial odometry using nonlinear optimization, The International Journal of Robotics Research 34(3)(2015) 314-334.

[5] J. Zhang, L. Tai, J. Boedecker, W. Burgard, M. Liu, Neural SLAM, in: Proc. 1st Conference on Robot Learning (CoRL 2017), 2017.

[6] Z. Zhang, A flexible new technique for camera calibration, IEEE Transactions on Pattern Analysis and Machine Intelligence, 22(2008) 1330-1334.

[7] G. Klein, D. Murray, Parallel tracking and mapping for small AR workspaces, in: Proc. ISMAR'07 Proceedings of the 2007 6th IEEE and ACM International Symposium on Mixed and Augmented Reality, 2007.

[8] L.V. Qiang, H. Lin, G. Wang, H. Wei, Y. Wang, ORB-SLAM-based tracing and 3D reconstruction for robot using Kinect 2.0, in: Proc. 2017 29th Chinese Control And Decision Conference (CCDC), 2017.

[9] E. Rublee, V. Rabaud, K. Konolige, G. Bradski, ORB: an efficient alternative to SIFT or SURF, in: Proc. 2011 IEEE International Conference on Computer Vision, 2011.

[10] B. Wu, ORB-SLAM2 source code explanation in detail[EB/OL]. <http://www.rosclub.cn/post-505.html>, 2016 (accessed 11.13.16).

[11] X. Gao, Non-linear optimization and g2o [EB/OL]. <http://www.rosclub.cn/post-245.html>, 2016 (accessed 10.25.16).

[12] S. Xie, W. Zhang, W. Ying, K. Zakim, Fast detecting moving objects in moving background using ORB feature matching, in: Proc. 2013 Fourth International Conference on Intelligent Control and Information Processing (ICICIP), 2013.

[13] R. Siegwart, I.R. Nourbakhsh, Introduction to Autonomous Mobile Robots, The MIT Press, Cambridge, 2004.

[14] W.N. Chai, C. Chen, Enhanced indoor navigation using fusion of IMU and RGB-D camera, in: Proc. International Conference on Computer Information Systems and Industrial Applications (CISIA 2015), 2015.

[15] S.-L. Sun, Z.-L. Deng, Multi-sensor optimal information fusion Kalman filter, Automatica 40(2004) 1017-1023.