A Novel Monocular SLAM Algorithm for High Real-Time Based on Kalman Filter

Wanqing Wu, Lin Ma, Bin Wang
School of Electronics and Information Engineering, Harbin Institute of Technology, Harbin, China
malin@hit.edu.cn

2021 International Wireless Communications and Mobile Computing (IWCMC) | 978-1-7281-8616-0/21/$31.00 ©2021 IEEE | DOI: 10.1109/IWCMC51323.2021.9498904

***Abstract*—Simultaneous Localization and Mapping (SLAM) has been a hot research direction in the field of mobile robots since it was proposed. ORB-SLAM is a typical algorithm in feature point SLAM system. However, because the ORB-SLAM can only provide sparse map construction, it can only be used for positioning. Just using ORB-SLAM to handle the positioning problem make the calculation complexity higher and reduce the real-time performance. Because of the complicated calculation, the ORB-SLAM is not practical to the embedded devices with low computing power. Therefore, we propose a monocular SLAM algorithm with high real-time performance based on Kalman filter. Our proposed algorithm reduces the time consumption of the system by modifying the posture optimization algorithm. Our algorithm provides accurate and efficient positioning for upper-level applications. We use Kalman filter modeling to optimize the pose of the camera. The experimental results show that compared with traditional algorithm, our proposed algorithm can effectively improve the real-time performance and reduces the time consumption of the system.**

Keywords—Kalman filter; SLAM; ORB; Pose optimization

I. INTRODUCTION

With the development of modern computer technology, mobile robots have become a very popular research field. Mobile robots are widely used in aviation, ocean exploration, military, medical, service and other fields [1]. Mobile robots have been applied to known environments, known partial environments and even unknown environments. [2] showed that the robot can generate the map and estimate the current position when exploring the random space, and Simultaneous Localization and Mapping (SLAM) is proposed to solve this problem. Compared with the data obtained by other sensors, the image data obtained by camera is compact, accurate, noninvasive and easy to understand. More importantly, visual sensors are inexpensive and ubiquitous. [3] showed that vision navigation has the advantages of large amount of information, high accuracy, small interference and strong real-time performance, so it is called a research hotspot. Therefore, visual sensors gradually replace other sensors and become the focus of SLAM research.

Nowadays, SLAM is often used for mobile robot positioning in indoor scenes. In 2015, [4] proposed the ORB-SLAM, it was proposed based on the feature point method. ORB-SLAM was used in monocular cameras. It was the pinnacle of feature point SLAM. It had excellent loop detection and correction. It was the first to use three parallel threads to complete SLAM, and the optimization algorithm was very good, which made ORB-SLAM better than others. The algorithm was much more robust. In 2017, [5] proposed ORB-SLAM2. ORB-SLAM2 added binocular camera and RGB-D camera modes, it expanded the scope of application.

The feature point method is to extract representative feature points from an image, such as Oriented fast and Rotated Brief (ORB), which was proposed in [6], Speeded Up Robust Feature (SURF), Scale Invariant Feature Transform (SIFT), etc. SIFT key point detection and feature vector description have been successfully applied to various situations where visual features are used, such as object recognition, image registration and splicing, and visual map construction. But it is limited due to the need for a lot of calculations, especially in real-time applications, such as visual odometry or low-cost devices like mobile phones. Therefore, there is a small computational cost, and the emergence of image pixel search algorithms for large amounts of dense data. ORB feature extraction is an efficient algorithm, which has the same feature matching performance as SIFT, and has strong image noise suppression performance. SIFT can be used in real-time systems. Therefore, ORB feature is the first option for our algorithm.

One of the current development directions of SLAM is to develop along the light weight, that is, to reduce calculation complexity and the time consumption of the system. SLAM is not only commonly used in computationally powerful devices, but is also being used in small embedded devices. [7-9] successfully applied SLAM to embedded devices, but they all have high requirements on the computing power of the devices. Using ORB-SLAM to handle the positioning problem will make the calculation complexity higher and reduce the real-time performance. Therefore, we propose a novel monocular SLAM algorithm for high real-time based on Kalman filter, which can improve the real-time performance of ORB-SLAM. Firstly, we perform ORB feature extraction and ORB feature matching on the image. Secondly, we use Kalman filter to model the camera's rotation vector and translation vector to optimize the camera pose. Finally, we select key frames and create map points. Experimental results show that compared with ORB-SLAM, our proposed algorithm can effectively reduce the time consumption of pose optimization and improves the real-time performance.

The remainder of this paper is organized as follows. In section II, we will introduce the monocular SLAM model based on Kalman filter proposed in our paper. And the problem formulation and solving algorithm of our algorithm will be demonstrated in Section III. In section IV, implementation and performance analysis will be provided. And conclusion will be drawn finally in section V.

II. System Model

Based on the ORB-SLAM algorithm, we propose a monocular SLAM algorithm based on Kalman filter. We use Kalman filter modeling to optimize the pose of the camera. Our proposed algorithm provides accurate and efficient positioning for upper-level applications. The positioning algorithm is to replace the original complex BA (Bundle Adjustment) optimization algorithm with a simple Kalman filter algorithm to reduce the amount of calculation. The overall architecture of the proposed algorithm in this paper is shown in Fig. 1.

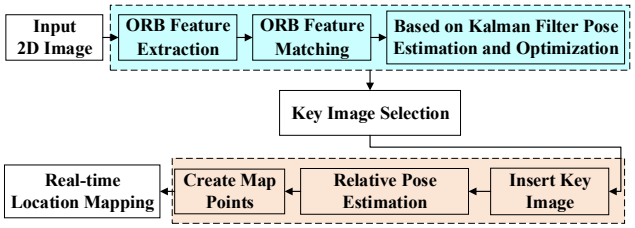

Fig. 1. Proposed Algorithm Framework

First, ORB feature extraction is carried out on the input two-dimensional image, and the initial feature points are obtained by feature matching with the previous frame image. Then we use Kalman filter to estimate the camera pose and optimize it. After obtaining the initially estimated camera pose and feature matching, a local visualization map is extracted from the common view of the key images maintained by the system. Re-projection method is used to retrieve the corresponding matching points in the current image and the local map. And then we use all matching point pairs to optimize the pose of the current camera. System decides whether to insert the current image as a new key image. Next, the inserted new key image is processed. The common view relationship is updated and the redundant map points are eliminated. The repeated map points are merged and new map points are added, and the key image connected to the current key image is optimized. Finally, the key images that do not meet the requirements are eliminated. Through the above algorithm, we realize the real-time positioning and mapping.

In summary, the core of our proposed algorithm is to use Kalman filter for optimal estimation in the pose estimation stage. We assume that the translation vector and the rotation vector are independent for each other. Performing Kalman filter on translation vector and rotation vector respectively. The prediction equation and update equation are constructed to calculate the optimal estimation of the pose at the next moment. The results of Kalman filter are used for real-time positioning and mapping in the system.

III. Proposed Method

A. Data Association Based on ORB Characteristics

To extract ORB features, we first detect Features from Accelerated Segment Test (FAST) key points. We take a pixel p in the image, and assume that its brightness is I_p. The surrounding 16 pixels centered on pixel p are shown in Fig. 2.

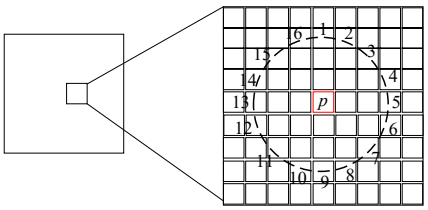

Fig. 2. FAST Key Points and Surrounding Pixel Representation

Set a threshold T. If the absolute value of the difference between the gray values of at least 3 points in p_1, p_5, p_9, p_{13} and the central pixel point p exceeds T, p is selected as the candidate key point, and then we make the next judgment. If p is selected as a candidate key point, and at least 9 consecutive points of the 16 points from p_1 to p_{16} have an absolute value of gray difference from the center p that exceeds T, then p is selected as the key point. We perform the non-maximum suppression processing on the current image. Then, select a range centered on the feature point p. If there are multiple feature points in this range, calculate the FAST score value (S value) for each feature point:

$$V = \max \begin{cases} \sum \left(p_{\text{pixel\_values}} - p \right) & \text{if } \left(p_{\text{value}} - p \right) > T \\ \sum \left(p - p_{\text{pixel\_values}} \right) & \text{if } \left(p_{\text{value}} - p \right) < T \end{cases}, \quad (1)$$

where p is the central pixel, $p_{\text{pixel\_values}}$ corresponds to the contiguous pixels in the circle and V represents the S value. We choose the key point with the maximum V of all feature points to be retained, and the remaining suppression. If there is only one key point in this range, keep it directly without calculating V.

ORB-SLAM extract a specific number of feature points for each image, but the number of extracted FAST key points is uncertain. Therefore, the selected FAST key points should be carefully selected, and the Harris response value of the extracted FAST key points should be calculated separately. Select the first N points with the largest response value as the candidate range of the final selected key points. ORB adds a description on the scale and rotation of FAST key points. Scale invariance is achieved by calculating the scale pyramid of the picture. The image is downsampled at different levels to obtain images of different resolutions, which is shown in Fig. 3. FAST key points are calculated at different scales.

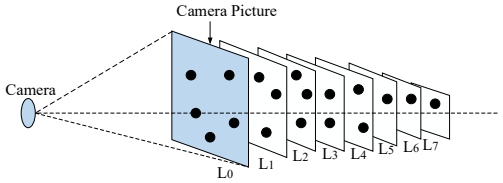

Fig. 3. Image Feature Extraction Pyramid

The description of feature rotation is realized by the gray-scale centroid method. In a small image block **B**, the moment of the image block is defined, the centroid of the image block is found through the moment, and the geometric center O and the

centroid of the image block are connected to obtain a vector $\overrightarrow{OC}$, we can calculate the direction of the feature point as:

$$m_{ij} = \sum_{x,y\in\mathbf{B}} x^i y^j I_{(x,y)} \qquad i,j=\{0,1\}, \tag{2}$$

$$C=\left(\frac{m_{10}}{m_{00}},\frac{m_{01}}{m_{00}}\right), \tag{3}$$

$$\theta=\arctan\left(\frac{m_{01}}{m_{10}}\right), \tag{4}$$

where (x,y) are the coordinates of the pixel relative to the feature point, m_{ij} is the moment of $\mathbf{B}$, C is the centroid of the image block and θ is the direction of the feature point.

Through the above method, FAST key points have a description of rotation and scale, which greatly improves the robustness of FAST corner points between different images. This improved FAST key point is called Oriented FAST key point. After the key points are extracted and filtered, a descriptor is calculated for each point. ORB uses an improved Binary Robust Independent Elementary Features (BRIEF) feature description. The improved BRIEF feature is also a binary descriptor, which adds rotation invariance to the BRIEF.

When calculating the BRIEF descriptor, in order to reduce noise, a Gaussian filter is performed on the image first. Take the feature point as the center and take the $S\times S$ window. Randomly select a pair of 5×5 sub-windows in the $S\times S$ window. We define test τ on patch $\mathbf{s}$ of size 5×5 as:

$$\tau(\mathbf{s};\mathbf{x},\mathbf{y})=\begin{cases}1, & p(\mathbf{x})<p(\mathbf{y})\\ 0, & p(\mathbf{x})\geq p(\mathbf{y})\end{cases}, \tag{5}$$

where $\mathbf{x}=(u,v)^{\mathrm{T}}$ and $\mathbf{y}=(u',v')^{\mathrm{T}}$ are random points. $p(\mathbf{x})$ and $p(\mathbf{y})$ are the sum of pixels of 5×5 sub-windows where points $\mathbf{x}$ and $\mathbf{y}$ are.

Randomly select N pairs of sub-windows in the $S\times S$ window and repeat the assignment to get a binary code:

$$f_n(\mathbf{s})\triangleq\sum_{1\leq i\leq n}2^{i-1}\tau(\mathbf{s};\mathbf{x}_i,\mathbf{y}_i), \tag{6}$$

where $f_n(\mathbf{s})$ is the descriptor of the feature point.

For any feature point, its BRIEF descriptor is a binary code of length N. This binary code is generated by N point pairs around the feature point, and these $2N$ points $(u_i,v_i),i=1,2,...,2N$ form a matrix $\mathbf{S}$:

$$\mathbf{S}=\begin{pmatrix}u_1 & u_2 & \dots & u_{2N}\\ v_1 & v_2 & \dots & v_{2N}\end{pmatrix}. \tag{7}$$

ORB uses the characteristic point direction θ and the corresponding rotation matrix $\mathbf{R}_\theta$ to construct the corrected $\mathbf{S}$:

$$\mathbf{S}_\theta=\mathbf{R}_\theta\mathbf{S}, \tag{8}$$

$$\mathbf{R}_\theta=\begin{bmatrix}\cos\theta & \sin\theta\\ -\sin\theta & \cos\theta\end{bmatrix}, \tag{9}$$

where θ is the direction of the feature points obtained above, and $\mathbf{S}_\theta$ is the corrected $\mathbf{S}$. The BRIEF descriptor is calculated by calculating the coordinates after the rotation:

$$g_n(\mathbf{s},\theta)\triangleq f_n(\mathbf{s})\big|(u_i,v_i)\in\mathbf{S}_\theta. \tag{10}$$

Next, we make the ORB feature matching. Feature matching is a more critical step in visual SLAM. It is able to determine the correspondence between the features observed in the current frame and the previously observed features. By accurately matching the descriptors between the image and the image or between the image and the map, a lot of burden can be reduced for the subsequent pose estimation and optimization. As mentioned earlier, the ORB feature descriptor BRIEF is a binary string. In ORB, a feature point is described by an *n*-bit descriptor. We take the feature points with small hamming distance as the best match of the current feature points. The descriptors of features $\mathbf{F}_1$ and $\mathbf{F}_2$ are:

$$\mathbf{F}_1=(x_1,x_2,...,x_n), \tag{11}$$

$$\mathbf{F}_2=(y_1,y_2,...,y_n). \tag{12}$$

The sum of the XOR values of the corresponding descriptors is defined as the Hamming distance of the descriptors, denoted by $D(\mathbf{F}_1,\mathbf{F}_2)$:

$$D(\mathbf{F}_1,\mathbf{F}_2)=\sum_{i=1}^{n}x_i\oplus y_i. \tag{13}$$

The smaller the $D(\mathbf{F}_1,\mathbf{F}_2)$, the higher the similarity between the two feature points. It can be seen that ORB feature matching only needs to perform the corresponding XOR operation on two binary strings of the same length. Then we can get the matching degree of the two feature points. The calculation is simple and can be completed in a short time.

B. Pose Optimization Algorithm Based on Kalman Filter

Kalman filter solves the weight relationship between the estimated value and the measured value to obtain a result closer to the true value. The result is called optimal estimation. Kalman filtering consists of two processes: prediction and update. In the prediction stage, the filter uses the estimation of the previous state to make a prediction of the current state. In the update phase, the filter uses the observed value of the current state to modify the predicted value obtained in the prediction phase. Then obtain a new estimate that is more in line with the true value.

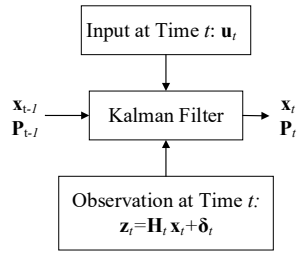

Fig. 4. Kalman Filter Process Representation

Fig. 4 is a schematic diagram of the Kalman filter. As

shown in Fig. 4, the initial state of the Kalman filter system is:

$$\mathrm{bel}(\mathbf{x}_0)=\mathcal{N}(\boldsymbol{\mu}_0,\mathbf{P}_0), \tag{14}$$

where $\mathcal{N}$ represents a normal distribution. $\boldsymbol{\mu}_0$ is the mean, and $\mathbf{P}_0$ is the variance.

The Kalman filtering process is divided into a prediction process and an update process. The first is the prediction process. The state transition model is a linear function:

$$\mathbf{x}_t=\mathbf{A}_t\mathbf{x}_{t-1}+\mathbf{B}_t\mathbf{u}_t+\boldsymbol{\varepsilon}_t. \tag{15}$$

Therefore, the conditional probability of state transition from $\mathbf{x}_{t-1}$ to $\mathbf{x}_t$ can be calculated as:

$$\mathrm{p}(\mathbf{x}_t\,|\,\mathbf{u}_t,\mathbf{x}_{t-1})=\mathcal{N}(\mathbf{A}_t\mathbf{x}_{t-1}+\mathbf{B}_t\mathbf{u}_t,\mathbf{R}_t). \tag{16}$$

According to the Bayesian equation, we calculate the distribution of the predicted state. At this time, all possible $\mathbf{x}_{t-1}$ need to be considered. The calculation equation is:

$$\overline{\mathrm{bel}}(\mathbf{x}_t)=\int \mathrm{p}(\mathbf{x}_t\mid\mathbf{x}_{t-1},\mathbf{u}_t)\mathrm{bel}(\mathbf{x}_{t-1})\mathrm{d}\mathbf{x}_{t-1}. \tag{17}$$

This is exactly the process of calculating the convolution of two Gaussian distributions:

$$\mathrm{p}(\mathbf{x}_t\,|\,\mathbf{u}_t,\mathbf{x}_{t-1})=\mathcal{N}(\mathbf{A}_t\boldsymbol{\mu}_{t-1}+\mathbf{B}_t\mathbf{u}_t,\mathbf{R}_t), \tag{18}$$

$$\mathrm{bel}(\mathbf{x}_{t-1})=\mathcal{N}(\boldsymbol{\mu}_{t-1},\mathbf{P}_{t-1}), \tag{19}$$

$$\begin{aligned}\overline{\mathrm{bel}}(\mathbf{x}_t)&=\int \mathrm{p}(\mathbf{x}_t\mid\mathbf{x}_{t-1},\mathbf{u}_t)\,\mathrm{bel}(\mathbf{x}_{t-1})\mathrm{d}\mathbf{x}_{t-1}\\&=\eta\int\exp\left\{-\frac{1}{2}\left(\mathbf{x}_t-\mathbf{A}_t\mathbf{x}_{t-1}-\mathbf{B}_t\mathbf{u}_t\right)^{\mathrm{T}}\mathbf{R}_t^{-1}\left(\mathbf{x}_t-\mathbf{A}_t\mathbf{x}_{t-1}-\mathbf{B}_t\mathbf{u}_t\right)\right\}\\&\quad\exp\left\{-\frac{1}{2}\left(\mathbf{x}_{t-1}-\mathbf{u}_{t-1}\right)^{\mathrm{T}}\mathbf{P}_t^{-1}\left(\mathbf{x}_{t-1}-\mathbf{u}_{t-1}\right)\right\}\mathrm{d}\mathbf{x}_{t-1}\end{aligned}, \tag{20}$$

$$\begin{aligned}\overline{\mathrm{bel}}(\mathbf{x}_t)&=\mathcal{N}(\bar{\mathbf{u}}_t,\bar{\mathbf{P}}_{t-1})\\&=\mathcal{N}\left(\mathbf{A}_t\boldsymbol{\mu}_{t-1}+\mathbf{B}_t\mathbf{u}_{t-1},\mathbf{A}_t\mathbf{P}_{t-1}\mathbf{A}_t^{\mathrm{T}}+\mathbf{R}_t\right)\end{aligned}. \tag{21}$$

Therefore, the prediction process of the Kalman filter is an analytical expression calculated based on the convolution of two Gaussian distributions. Next is the update process. Assuming that the observation equation is also a linear equation, and the noise is Gaussian noise. The observation equation is:

$$\mathbf{z}_t=\mathbf{H}_t\mathbf{x}_t+\boldsymbol{\delta}_t. \tag{22}$$

The conditional probability of $\mathrm{p}\left(\mathbf{z}_t\,|\,\mathbf{x}_t\right)$ is calculated by linear transformation of Gaussian distribution:

$$\mathrm{p}\left(\mathbf{z}_t\,|\,\mathbf{x}_t\right)=\mathcal{N}\left(\mathbf{H}_t\mathbf{x}_t,\mathbf{Q}_t\right). \tag{23}$$

This is exactly the problem of the product of two Gaussian distributions:

$$\mathrm{p}(\mathbf{z}_t\mid\mathbf{x}_t)=\mathcal{N}\left(\mathbf{z}_t;\mathbf{H}_t\mathbf{x}_t,\mathbf{Q}_t\right), \tag{24}$$

$$\overline{\mathrm{bel}}(\mathbf{x}_t)=\mathcal{N}\left(\mathbf{x}_t;\bar{\mathbf{u}}_t,\bar{\mathbf{P}}_t\right), \tag{25}$$

$$\begin{aligned}\mathrm{bel}(\mathbf{x}_t)&=\eta\,\mathrm{p}(\mathbf{z}_t\mid\mathbf{x}_t)\overline{\mathrm{bel}}(\mathbf{x}_t)\\&=\eta\exp\left\{-\frac{1}{2}\left(\mathbf{z}_t-\mathbf{H}_t\mathbf{x}_t\right)^{\mathrm{T}}\mathbf{Q}_t^{-1}\left(\mathbf{z}_t-\mathbf{H}_t\mathbf{x}_t\right)\right\}\\&\quad\exp\left\{-\frac{1}{2}\left(\mathbf{x}_{t-1}-\bar{\mathbf{u}}_t\right)^{\mathrm{T}}\bar{\mathbf{P}}_t^{-1}\left(\mathbf{x}_{t-1}-\bar{\mathbf{u}}_t\right)\right\}\end{aligned}. \tag{26}$$

Therefore, based on the method of finding the distribution of the product of Gaussian variables, it can be derived that the result is still the Gaussian distribution, and its second moment is expressed as:

$$\mathrm{bel}\left(\mathbf{x}_t\right)=\mathcal{N}\left(\bar{\mathbf{u}}_t+\mathbf{K}_t(\mathbf{z}_t-\mathbf{H}_t\bar{\mathbf{u}}_t),(\mathbf{I}-\mathbf{K}_t\mathbf{H}_t)\bar{\mathbf{P}}_t\right). \tag{27}$$

The Kalman filter gain is:

$$\mathbf{K}_t=\bar{\mathbf{P}}_t\mathbf{H}_t^{\mathrm{T}}\left(\mathbf{H}_t\bar{\mathbf{P}}_t\mathbf{H}_t^{\mathrm{T}}+\mathbf{Q}_t\right)^{-1}, \tag{28}$$

where $\mathbf{x}_t$ is the optimal estimation of Kalman filter.

The motion model constructed in our algorithm is assumed to have a constant velocity and constant angular velocity model, which means that the motion velocity between two adjacent frames is constant. This does not mean that it is assumed that the camera is always moving at a constant speed. But the statistical model of motion in the time step in our algorithm is on average expected to use a Gaussian distribution to represent undetermined acceleration. The ORB feature matching is used to find more matching point pairs between the current frame and the local map before the pose optimization, set to n. The PnP algorithm is used to solve the pose. The PnP algorithm is an algorithm that calculates the current camera pose by matching 2D-3D point pairs. The PnP algorithm can be used to calculate the spatial attitude of the camera, and it can also be used for spatial positioning.

We believe that the rotation angle and the translation vector are independent of each other, so two identical Kalman filter models are established. The process noise and the observation noise both obey the Gaussian distribution with zero mean, and the variances are $\mathbf{R}$ and $\mathbf{Q}$ respectively. The first is the rotation vector Kalman filter modeling. The rotation vector $\mathbf{r}$ solved by PnP is used as the predicted value of the Kalman filter, and the classical Kalman filter equation is used to calculate the covariance matrix:

$$\bar{\mathbf{r}}_k=\mathbf{r}'_k, \tag{29}$$

$$\bar{\mathbf{P}}_k=\mathbf{P}_{k-1}+\mathbf{R}. \tag{30}$$

The Kalman filter gain is calculated by the predicted value of the rotation vector and the predicted value of the covariance matrix. The previous rotation vector (considered the true value of the previous moment), the predicted value of the current rotation vector and the Kalman filter gain are used to calculate the rotation vector at the current moment:

$$\mathbf{K}_k=\bar{\mathbf{P}}_k\left(\bar{\mathbf{P}}_k+\mathbf{Q}\right)^{-1}, \tag{31}$$

$$\mathbf{r}_k=\mathbf{r}_{k-1}+\mathbf{K}_k\left(\bar{\mathbf{r}}_k-\mathbf{r}_{k-1}\right), \tag{32}$$

$$\mathbf{P}_k = \left(\mathbf{I} - \mathbf{K}_k\right)\overline{\mathbf{P}}_k . \tag{33}$$

In the same way, the translation vector **t** is modeled. Taking the translation vector solved by PnP as the predicted value of the Kalman filter, and bring it into the classical Kalman filter equation to calculate the covariance matrix:

$$\overline{\mathbf{t}}_k = \mathbf{t}'_k , \tag{34}$$

$$\overline{\mathbf{P}}_k = \mathbf{P}_{k-1} . \tag{35}$$

Calculate the Kalman filter gain through the predicted value of the translation vector and the predicted value of the covariance matrix. Use the translation vector at the previous moment (considered the true value of the previous moment), the predicted value of the translation vector at the current moment and the Kalman filter gain Value calculate the translation vector at the current moment:

$$\mathbf{K}_k = \overline{\mathbf{P}}_k \left(\overline{\mathbf{P}}_k + \mathbf{Q}\right)^{-1}, \tag{36}$$

$$\mathbf{t}_k = \mathbf{t}_{k-1} + \mathbf{K}_k \left(\overline{\mathbf{t}}_k + \mathbf{t}_{k-1}\right), \tag{37}$$

$$\mathbf{P}_k = \left(\mathbf{I} - \mathbf{K}_k\right)\overline{\mathbf{P}}_k . \tag{38}$$

In this section, we complete the Kalman filter modeling of the rotation vector and the shift vector, construct the prediction equation and update equation. They are used to calculate the optimal estimation of the pose at the next moment.

IV. Implementation And Performance Analysis

A. Experiment Setup

The system used in our paper is Ubuntu 16.04 version, the experimental platform is ROS Kinetic Kame, and the camera type used is a monocular camera. In order to verify the feasibility of the algorithm we proposed, it is necessary to select a suitable experimental scene for testing. The experimental environment of our paper is K1928 laboratory in the Science and Technology Innovation Building of Harbin Institute of Technology. The environment we choose has many objects, which is convenient for feature point selection. And the experimental scene contains loops which is convenient to test the loopback detection. Therefore, this experimental environment is suitable for verifying the algorithm we proposed in this paper. Fig. 5 is a plan view of the experimental scene, and the red arrow is the path of the map.

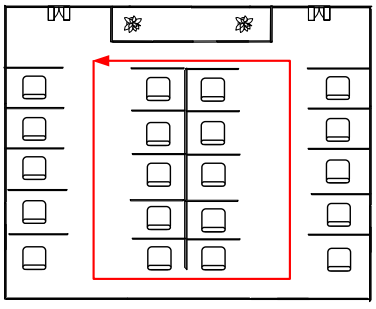

Fig. 5. Floorplan of Experimental Scene

B. Experiment Results

We study the experiment of ordinary monocular camera, and the input raw data is the image of the external environment directly obtained by the monocular camera. ORB feature extraction and matching are performed on the input images using the method described above. Fig. 6 shows the ORB feature extraction results of the images. Based on this, ORB feature matching is performed for the two images. Fig. 7 shows the results of original ORB feature matching point pairs, and Fig. 8 shows the results of filtered ORB feature matching point pairs. Fig. 8 shows that the filtered feature point pair is more accurate than the original matching point pair.

Fig. 6. ORB Feature Extraction Results

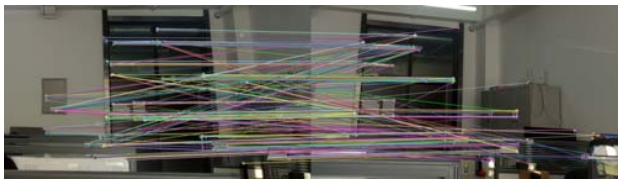

Fig. 7. The Unfiltered ORB Features Match Point Pairs

Fig. 8. The Filtered ORB Features Match Point Pairs

The key points and descriptors of the feature points are extracted through the algorithm described above. The extracted features are represented as green dot boxes in Fig. 9. After extracting the ORB feature of the current frame, the extracted feature points are matched with the initialized map points, and the current camera pose is estimated. The Kalman filter model established above is used to optimize the current camera pose. Finally, it is determined whether to store the current frame as a key image.

Fig. 9. Grayscale and ORB Features

The map point is the 3D position corresponding to the feature point in the real environment. All map points form a sparse map, which is the mapping part in SLAM. In Fig. 10, it is represented by red points and black points, where red represents the map point observed in the current frame (that is, the map point that matches the features extracted from the image), and the black point is other map points. Repeated movement around the object, then the map points will be enough, we can see the outline of the object. The selected key image is represented as a blue frame in Fig. 10, and the green frame is the current camera location. The final pose obtained after a series of optimizations on the pose of the key image is the pose shown in the Fig. 10. The pose of all the key images can form a global trajectory, which is represented as a green line connecting the key images in the Fig. 10.

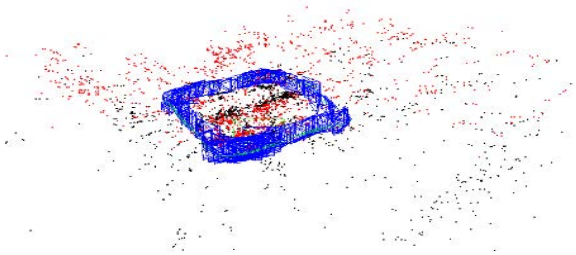

Fig. 10. Map Cloud Points and Key Images

Data support is needed to determine the experimental effect of algorithm modification. The running time of the ORB-SLAM pose optimization program using Kalman filter and BA optimization algorithm is compared, and the average value is calculated. As shown in Fig. 11, the light blue point is the running time of the Kalman filter, and the red line is the mean running time; The rose red point is the running time of algorithm in [5], and the red line is the mean running time. TABLE I lists the comparison of the mean running time of the algorithm we proposed and the algorithm in [5].

TABLE I: Comparison of the Mean Running Time

Algorithm	**Mean Running Time /ms**
Proposed Algorithm	2.149
Algorithm in [5]	7.208

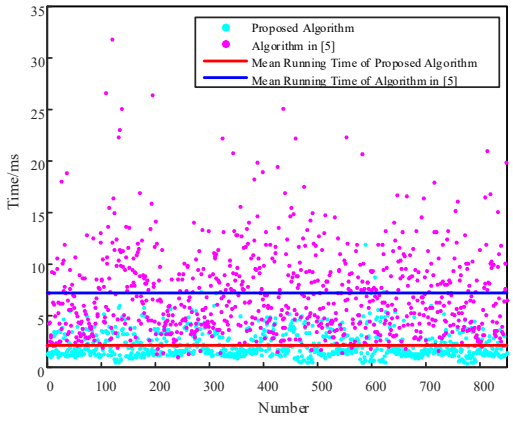

Fig. 11. Comparison of the Mean Running Time

V. Conclusion

In this paper, we propose a novel monocular SLAM algorithm for high real-time based on Kalman filter. Through the comparison of the experimental results, it can be seen that using Kalman filter to optimize the pose significantly reduces the amount of calculation of the algorithm, and reduces the running time of this part of the algorithm to 30% of the original. It can be concluded that using Kalman filter to optimize the pose in ORB-SLAM can obviously reduce the running time of the system. Compared with the original ORB-SLAM system, we improve the real-time performance of SLAM positioning and mapping.

Acknowledgment

This paper is supported by National Natural Science Foundation of China (61971162, 41861134010) and Aeronautical Science Foundation of China (2020Z066015002).

References

[1] C. H. A. H. B. Baskoro, H. M. Saputra, M. Mirdanies, V. Susanti, M. F. Radzi and R. I. A. Aziz, "An Autonomous Mobile Robot Platform for Medical Purpose," 2020 International Conference on Sustainable Energy Engineering and Application (ICSEEA), Tangerang, Indonesia, 2020, pp. 41-44.

[2] S. -W. Kang, S. -H. Bae and T. -Y. Kuc, "Feature Extraction and Matching Algorithms to Improve Localization Accuracy for Mobile Robots," 2020 20th International Conference on Control, Automation and Systems (ICCAS), Busan, Korea (South), 2020, pp. 991-994.

[3] W. Wei, L. Tan, G. Jin, L. Lu and C. Sun, "A Survey of UAV Visual Navigation Based on Monocular SLAM," 2018 IEEE 4th Information Technology and Mechatronics Engineering Conference (ITOEC), Chongqing, China, 2018, pp. 1849-1853.

[4] R. Mur-Artal, J. M. M. Montiel and J. D. Tardós, "ORB-SLAM: A Versatile and Accurate Monocular SLAM System," in IEEE Transactions on Robotics, vol. 31, no. 5, pp. 1147-1163, Oct. 2015.

[5] R. Mur-Artal and J. D. Tardós, "ORB-SLAM2: An Open-Source SLAM System for Monocular, Stereo, and RGB-D Cameras," in IEEE Transactions on Robotics, vol. 33, no. 5, pp. 1255-1262, Oct. 2017.

[6] E. Rublee, V. Rabaud, K. Konolige and G. Bradski, "ORB: An efficient alternative to SIFT or SURF," 2011 International Conference on Computer Vision, Barcelona, 2011, pp. 2564-2571.

[7] E. Tang, S. Niknam and T. Stefanov, "Enabling Cognitive Autonomy on Small Drones by Efficient On-Board Embedded Computing: An ORB-SLAM2 Case Study," 2019 22nd Euromicro Conference on Digital System Design (DSD), Kallithea, Greece, 2019, pp. 108-115,.

[8] D. Nguyen, A. Elouardi, S. A. Rodriguez Florez and S. Bouaziz, "HOOFR SLAM System: An Embedded Vision SLAM Algorithm and Its Hardware-Software Mapping-Based Intelligent Vehicles Applications," in IEEE Transactions on Intelligent Transportation Systems, vol. 20, no. 11, pp. 4103-4118, Nov. 2019.

[9] S. Aldegheri, N. Bombieri, D. D. Bloisi and A. Farinelli, "Data Flow ORB-SLAM for Real-time Performance on Embedded GPU Boards," 2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Macau, China, 2019, pp. 5370-5375.

[10] S. Yang and S. Scherer, "Monocular Object and Plane SLAM in Structured Environments," in IEEE Robotics and Automation Letters, vol. 4, no. 4, pp. 3145-3152, Oct. 2019.

[11] B. Han and L. Xu, "A Monocular SLAM System with Mask Loop Closing," 2020 Chinese Control And Decision Conference (CCDC), Hefei, China, 2020, pp. 4762-4768.

[12] T. -S. Lou, H. -Y. Ban, S. -N. Zhao, Z. -D. He and Y. Wang, "Rank Kalman Filter-SLAM for Vehicle with Non-Gaussian Noise," 2020 5th International Conference on Advanced Robotics and Mechatronics (ICARM), Shenzhen, China, 2020, pp. 12-15.