Journal of Intelligent & Robotic Systems (2018) 92:107–124
https://doi.org/10.1007/s10846-017-0742-z

Critical Rays Self-adaptive Particle Filtering SLAM

Wenjie Song[1,2] · Yi Yang[1] · Mengyin Fu[1,3] · Alain Kornhauser[2] · Meiling Wang[1]

Received: 9 October 2016 / Accepted: 8 November 2017 / Published online: 2 January 2018

Abstract
This paper presents CRSPF-SLAM, a critical rays self-adaptive particle filtering occupancy grid based SLAM system that can operate efficiently with different kinds of odometer in real time, in small and large, indoor and outdoor environments for various platforms. Its basic idea is to eliminate the accumulated error of odometer through scan to map matching based on particle filtering. Through some improvements for the original particle filtering method, the lidar system becomes more robust to conduct accurate localization and mapping. Specifically, in our proposed method, particle filter based on Monte-Carlo algorithm is designed to be out-of-step to the odometer; During the scan matching process, the influence of some critical rays selected through a ray-selection algorithm is enhanced and that of the unreliable rays is weaken or removed; The current optimal match value is regarded as the feedback to reset the particle number and the filtering range; Once the optimal pose and scan are obtained, the previous error scan stored in the map will be removed. It is also introduced in the paper that the method can work effectively with dead reckoning, visual odometry and IMU, respectively. And we have tried to use it on different types of platforms — an indoor service robot, a self-driving car and an off-road vehicle. The experiments in a variety of challenging environments, such as bumpy and characterless area, are conducted and analyzed.

Keywords SLAM · Particle filtering · Critical rays · Self-adaptive · Occupancy grid

1 Introduction

In recent years, autonomous robots with different functions start to get involved and play an essential role in people's daily life. To help an autonomous robot to survive and navigate within its surroundings, two basic and critical problems need to be addressed: Mapping the environment and finding its relative pose within the map. For this reason, simulataneous localization and mapping (SLAM) [17] undoubtedly is the most important component for an autonomous robot, whose basic idea is locating the robot by matching its observations with a consistent and incremental map and updating the map at the same time. With the development of robot technology, a wealth of research into the SLAM problem has been conducted, resulting in many reliably working solutions for indoor service robot, outdoor detecting robot and so on. Overall, they can be roughly divided into three types according to the sensors: laser range finder(LRF), camera and RGB-D sensor. What's more, the core algorithms of them generally include three different kinds of optimization methods: KF [2], PF [6] and g2o [16].

Among all the sensors, laser range finder has been the most widely used with occupancy grid map on account of its high accuracy and reliability. Many researchers have realized accurate positioning and mapping for the autonomous robot in different situations by using laser solely or combining it with odometry, which has been studied, e.g. in [7, 9, 12, 15, 24–26] and so on. For this type of methods, the core part is the scan-matching approach, which can be scan

Electronic supplementary material The online version of this article (https://doi.org/10.1007/s10846-017-0742-z) contains supplementary material, which is available to authorized users.

✉ Yi Yang
yang_yi@bit.edu.cn

Wenjie Song
wenjiebit@gmail.com

1 School of Automation, Beijing Institute of Technology, 625 room, 6th building, Beijing, China

2 Princeton Autonomous Vehicle Engineering, Princeton University, Princeton, NJ, USA

3 Nanjing University of Science and Technology, Nanjing, China

to each other matching and scan to map matching. Furthermore, it also can be categorized into three types: feature to feature, point to feature and point to point. In the feature to feature matching approaches, features such as range extrema [18], corners or lines [13] extracted from laser scans are matched. Similarly, the second approach matches current points to the detected features in the last frame. Compared with the two types, the third type is widely used and plenty of methods are proved to be highly reliable. Some of them are conducted based on scan to each other matching, and the others are based on scan to map matching. The representative works are iterative closest point (ICP) [22], iterative matching range point (IMRP) [19] and the popular iterative dual correspondence (IDC) [1]. These methods always can work well in the static or ideal environments, but get into trouble in the noisy or real-life environments. What's more, when there are few scans or the scan feature is not obvious, they almost can't work normally and their real-time capability goes to bad when there are too many rays. To improve them, [7] avoids searching for point associations by simply matching points with the same bearing in the polar form to develop a faster scan-matching algorithm, but they only conduct simulation in the static environment in their paper and the dynamic or difficult environment seems challenging for it. Taleghani et al. [25] proposes a innovative method named ICE matching, which detects novel feature points, such as intersection, corner and end of wall, from each frame of scan and conducts scan matching focusing on the detected points. CRSM-SLAM [26] employs Random Restart Hill Climbing (RRHC) method to conduct point to map scan matching, which also extracts feature points as critical rays for its matching to reduce the complexity and time. These two methods extract features from scans but they don't belong to type one or type two, in other words, they only use the feature points to make the scan matching more robust. Inspired by them, we enhance the influence of some rays which are called critical rays during our scan matching. We require that the critical rays must can represent the architectural feature of the scan and the number of them is as little as possible. We have tested the open source of these two methods and found their critical rays selection method can't meet our requirements. Finally, we succeed to modify IPAN [3] algorithm to select critical rays for our system, and compared it with the two selection methods on different environments. On the other hand, many researchers apply particle filtering or kalman filtering to conduct scan matching, such as Gmapping [12] and DP-SLAM [9]. They usually can obtain perfect results even in some tough environments according to the simulation with robot's moving slowly, but their real-time performance sometimes can't make it for the practical application such as robot's moving fast and moving in the bumpy environment. To obtain a better applicability, TinySLAM [24] simplifies the particle filter algorithm to get a good real-time capacity and a not bad result. They eliminate positioning error of the odometry by using a simple Monte-Carlo algorithm to match the current scan with the map. However, in their method, the filtering range and particles number is constant and the process of dead reckoning is isochronous with particle filtering, which leads to its poor robustness in some cases. Our method adopts and improves their scan matching method and designs a feedback according to the last match value to change the filtering range and particles number in real time. Furthermore, in our method, the execution frequency of odometer's dead reckoning is designed to be much higher than filtering process to make the system more robust and another thread is created to try to remove the inaccurate scans or noises drawn in the map wrongly according to the high-reliable frames.

In addition to LRF, mono camera and rgbd sensors are more and more popular for the SLAM researcher. Both LSD-SLAM [10] and ORB-SLAM [21] are the excellent mono SLAM methods, which can create a sparse 3D map. Nevertheless, this kind of method can't recover absolute scale without some kind of external reference. Therefore, more researchers try to use rgbd sensors or stereo camera to help the robot to locate itself and create a absolute scale map, such as [8, 14] and [11]. In our paper, we also have tried to combine the visual odometry of [11] with our lidar filtering system instead of the wheel odometry to get a better result.

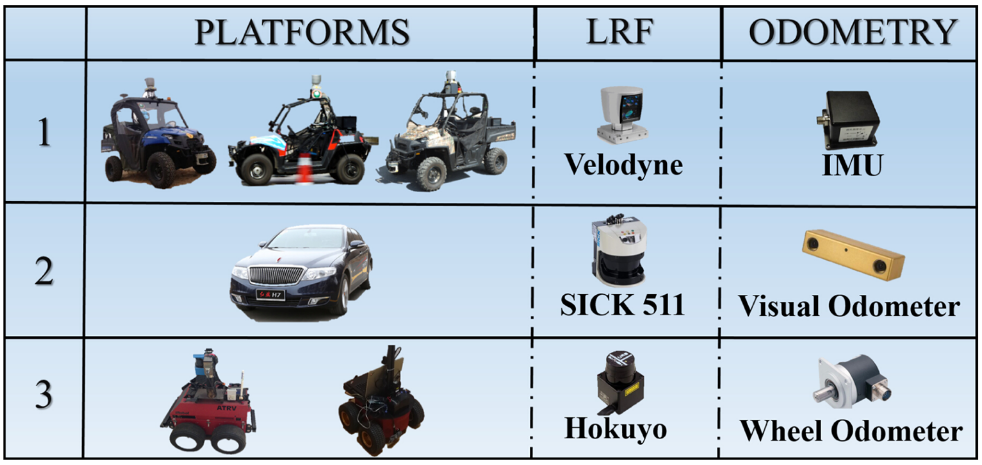
Of course, in some special application scenarios, researchers also use other kinds of sensors to find landmarks for locating the robot and creating the map. For example, [27] uses the wireless sensor nodes as landmarks in their method to help the robot work indoor.

The contribution of this paper is that we present a real-time and robust SLAM method for practical applications in diverse environments. We design a simple particle filtering to conduct scan matching, and the previous match values are regarded as feedback to optimize the filter at the same time. What's more, to strengthen the structural characteristics of the environment, some critical rays are selected from the scan based on computation geometry. On the other hand, we try to combine it with different kind of odometry and the handle rate of dead reckoning is designed to be much higher than scan matching. In our paper, experiments with wheel odometry, visual odometry and IMU are conducted on different kinds of platforms, respectively. Specifically, different platform employs different odometry and LRF sensor with different data processing method. Moreover, the matched high-reliable scans are used to remove the inaccurate scans or noises drawn in the map wrongly before so that the map is able to adapt to the changing environment.

2 System Overview

We have developed several groups of experimental platforms to demonstrate the effectiveness of our system, which are shown in Fig. 1. In the first group, the autonomous all-terrain vehicles — IN^2BOT series, which are modified by ourselves, mainly equip a Velodyne 3D lidar as the laser range finder and an IMU as the odometry. Specifically, information from 3D lidar is processed into the format of 2D occupancy grid according to an improved ground model algorithm and the velocity of car from IMU is accumulated for dead reckoning. They have taken part in "China Intelligent Vehicle Future Challenge" for many times and gotten good grades in the last few years. Generally, this group is utilized to test our methods in the bumpy road or the field environments. In the second group, a HongQi car is used to test our methods in the high speed situation. For the SLAM system, we mainly add a SICK 2D lidar for obstacle detecting and a Bumblebee2 stereo camera for visual odometer to this car. The detection range of this 2D lidar is up to 80 m outdoor and the baseline of the cameras is 12 cm. The third group is primarily for indoor service, that includes two small robots: iRobot-ATRV and Pioneer3. On the one hand, they mainly equip a Hokuyo (URG-04LX-UG01) 2D lidar as their LRF, which has a detection range of 5 m and a view angle of 180°. On the other hand, there are two high-precision encoders on their wheels to assist them to realize the wheel odometer function. In general, the proposed CRSPF-SLAM method has been tested on each group in different kinds of environments and the experimental results are discussed in this paper.

Figure 2 shows the software architecture of our method in this paper. It includes three main modules: Particle Filtering, Odometry and Map Update, that are operating at the same time in three different threads. Moreover, they share some essential information to each other. Specifically, Odometry module gives its current pose to PF incessantly as the observed or reference pose for scan matching. In return, PF module gives back the obtained accurate pose to Odometry as its new initial pose under certain conditions. And on the same conditions, PF module also offers the optimal pose and scan to Map Update module to refresh the map. Meanwhile, Odometry module is sending its current pose to Map Update continually for robot's location in the map. More specifically, firstly, the algorithm performs initialization of the structures needed to store the map, all acquired scans and other information. Then, for PF module, there are three main sub-parts. Rays Selection will process the original scan if there is a new frame of laser data to obtain the critical rays for enhancing the environmental structure characteristics. Then, Scan Matching matches the processed scan with the map based on the observed pose from Odometry module to get an optimal pose and a match value. Then the current match value will be used as feedback to reselect a suitable filtering range and particles number in the next frame. Furthermore, the latest several continuous match values are analyzed in Match Values Analyzer to decide whether the "optimal pose" can be sent to Odometry module and what will be done in Map Update module. For Odometry module, only one of the three odometers(wheel odometer, visual odometer and IMU) is used to conduct dead reckoning for current rough pose. And for Map Update module, there are three types of modifications: Draw Scan, Remove Error and Remain Unchanged. When Match Values Analyzer gives a "good result", the current scan with the optimal pose is drawn on the map and the error on the map is removed at the same time. However, the map remains unchanged with the observed pose from Odometry when



	PLATFORMS	LRF	ODOMETRY
1		Velodyne	IMU
2		SICK 511	Visual Odometer
3		Hokuyo	Wheel Odometer

Fig. 1 Experimental platforms that CRSPF-SLAM works on

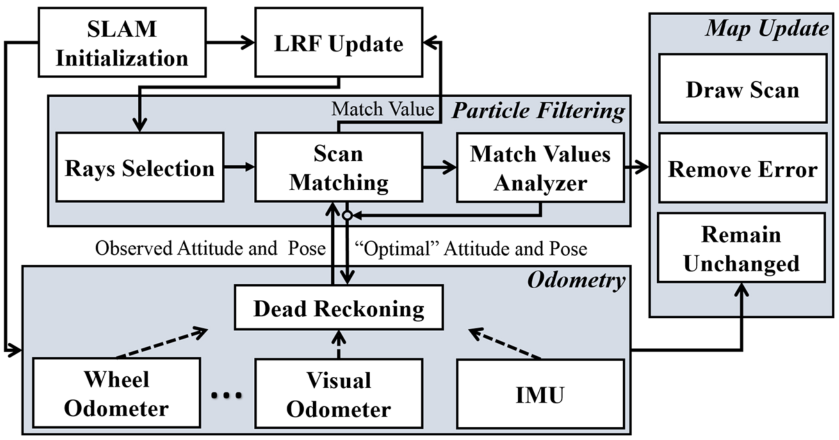


Fig. 2 Software architecture of CRSPF-SLAM

Match Values Analyzer generates a "bad result". In this way, we can always get a stable map and a not bad location in as many cases as possible.

3 Odometry

In the Odometry module, we have tested three different kinds of odometers on various types of platforms. Their common goal is to obtain a rough but real-time robot's pose as the observation$\{\hat{x}, \hat{y}, \hat{\omega}\}$.

3.1 Wheel Odometer

In our research, we adopt two motion models of mobile robot to conduct dead reckoning, respectively. For some robots like that in Fig. 3a, they use two actual drive wheels with two encoders to move and generate odometry measurements, which is discussed in [4]. When the robot moves straight, its model is shown in formula (1). Otherwise, the motion model is shown in formula (2) according to Fig. 3b, where, $r_k = \frac{B}{2}(\frac{L_k+R_k}{L_k-R_k})$, L_k and R_k are the walking ranges of the two driving wheels, and B is the wheel span. For other robots like that in Fig. 4a, Ackerman Principle is usually used to get a current rough pose, which is shown in formula (3) according to Fig. 4b, where, V_k is the current speed of the driving wheel, Φ_k is the current steering angle and L is the wheel base.

$$\begin{bmatrix} \hat{x}_k \\ \hat{y}_k \\ \hat{\omega}_k \end{bmatrix} = \begin{bmatrix} \hat{x}_{k-1} \\ \hat{y}_{k-1} \\ \hat{\omega}_{k-1} \end{bmatrix} + \begin{bmatrix} \frac{R_k+L_k}{2} cos\hat{\omega}_{k-1} \\ \frac{R_k+L_k}{2} sin\hat{\omega}_{k-1} \\ 0 \end{bmatrix} \tag{1}$$

$$\begin{bmatrix} \hat{x}_k \\ \hat{y}_k \\ \hat{\omega}_k \end{bmatrix} = \begin{bmatrix} \hat{x}_{k-1} \\ \hat{y}_{k-1} \\ \hat{\omega}_{k-1} \end{bmatrix} + \begin{bmatrix} r_k \left[sin\hat{\omega}_{k-1} - sin(\hat{\omega}_{k-1} + \frac{R_k-L_k}{B}) \right] \\ r_k \left[cos(\hat{\omega}_{k-1} + \frac{R_k-L_k}{B}) - cos\hat{\omega}_{k-1} \right] \\ \frac{R_k-L_k}{B} \end{bmatrix} \tag{2}$$

$$\begin{bmatrix} \hat{x}_k \\ \hat{y}_k \\ \hat{\omega}_k \end{bmatrix} = \begin{bmatrix} \hat{x}_{k-1} \\ \hat{y}_{k-1} \\ \hat{\omega}_{k-1} \end{bmatrix} + \begin{bmatrix} V_k \cdot sin\hat{\omega}_{k-1} \\ V_k \cdot cos\hat{\omega}_{k-1} \\ (V_k \cdot tan\Phi_{k-1})/L \end{bmatrix} \tag{3}$$

3.2 Visual Odometer

For Hongqi car, we use the visual odometry algorithm proposed in [11], which can obtain a rough pose through conducting feature circle-matching and EKF-based ego-motion estimation as shown in Fig. 5.

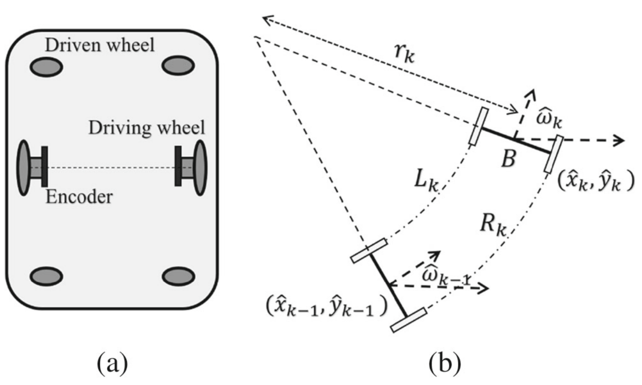


Fig. 3 Differential steering robot and its odometer model

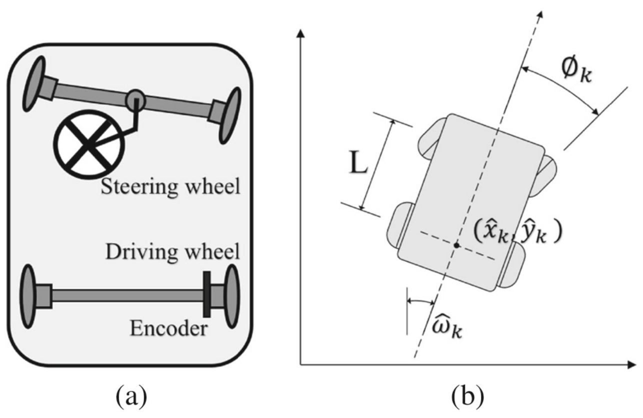


Fig. 4 Servo steering robot and its odometer model

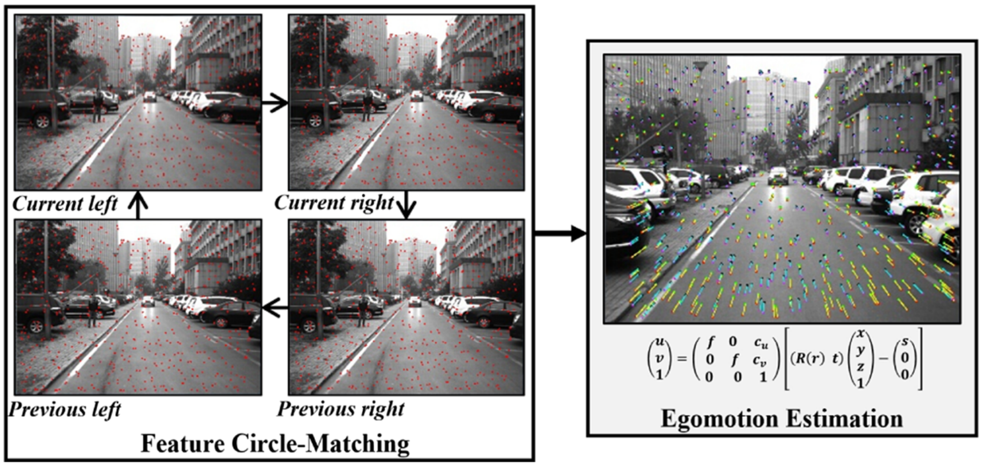

Current left
Current right
Previous left
Previous right
Feature Circle-Matching
$\begin{pmatrix} u \\ v \\ 1 \end{pmatrix} = \begin{pmatrix} f & 0 & c_u \\ 0 & f & c_v \\ 0 & 0 & 1 \end{pmatrix} \left[(R(r)\ \ t) \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} - \begin{pmatrix} s \\ 0 \\ 0 \end{pmatrix} \right]$
Egomotion Estimation

Fig. 5 Stereo visual odometer used in our system

3.3 IMU

We also tried to use IMU to accumulate robot's movement in the field environments. As we all know, IMU has similar properties with odometer. Their measuring error increases as time goes on but the results are always continuous. What's more, IMU has higher robustness in the bumpy environment than other odometers, which is the main reason why we choose IMU for IN^2BOTs.

4 Particle Filtering

In CRSPF-SLAM method, the Particle Filtering module is designed to eliminate the accumulative error of the Odometry module. The input source of this module is the real-time and stable scan data (scanning spots) from LRF Update part and an observed pose from the Odometry module. Its outputs include the analysis result of the matching status and an accurate pose with the corresponding scan.

4.1 LRF Update

When the robot or autonomous vehicle running indoor or on the flat roads, 2D lidars such as SICK and Hokuyo usually can meet the detection requirements. For these platforms or cases, the original lidar data can be directly used as the scan in our system. Nonetheless, when working in the rugged environments, there is so much noise caused by turbulence in the original data that SLAM is hardly able to work effectively. For this reason, we choose a 3D lidar for IN^2BOTs to work in the bumpy environments so that more environmental information can be utilized to overcome the effects of turbulence.

In our SLAM system, we adopt a modified method based on our previous research [28] to process the enormous points cloud data of the Velodyne lidar to get the real-time and stable scan data source with the same format as 2D lidar for SLAM. Thus, we can not only improve the robustness of detection, but also avoid dealing with the 3D points cloud directly to reduce the time-consumption of this part immensely. According to our previous method, the 3D points cloud (Fig. 6a) is divided into 720 sectors, each sector stands for 0.5°, which is shown in Fig. 6b. For each sector, the points in it are sorted to generate a Height-Range(HR) graph as Fig. 6d. We used to calculate the height difference and the slope between consecutive points for finding the obstacle point to get the range of this sector. If both of these two values exceed some predefined thresholds(10 cm in the height and 15 degrees in the slope), then the first point is set to the obstacle point. As we explained in our previous paper [28], this method succeeded to work efficiently to obtain the passable area (720 obstacle points) for path planning even in the bumpy field. Even so, we found the obtained long-distance obstacle points are not stable enough for scan matching, which causes a lot of noise in the SLAM map. So we only use the obstacle points that are less than 20 meters as the scan data source for the SLAM system, which are the red points in Fig. 6c.

4.2 Critical Rays Selection

In the vast majority of cases, enough stable scan points can be obtained from LRF Update module. However, accurate location in the longitudinal direction is extremely difficult for lidar SLAM in the environments with few structural characteristics, such as the long-straight corridor(Fig. 7a and b), expressway and so on. To solve this problem, we select critical rays or dominant points from current scan points to increase structural features so that the scan matching in the longitudinal direction can be more accurate, which is shown in Fig. 7c and d. As we can see, the weeny characteristics such as the closed doors' frame and natural gas pipeline can be detected easily.

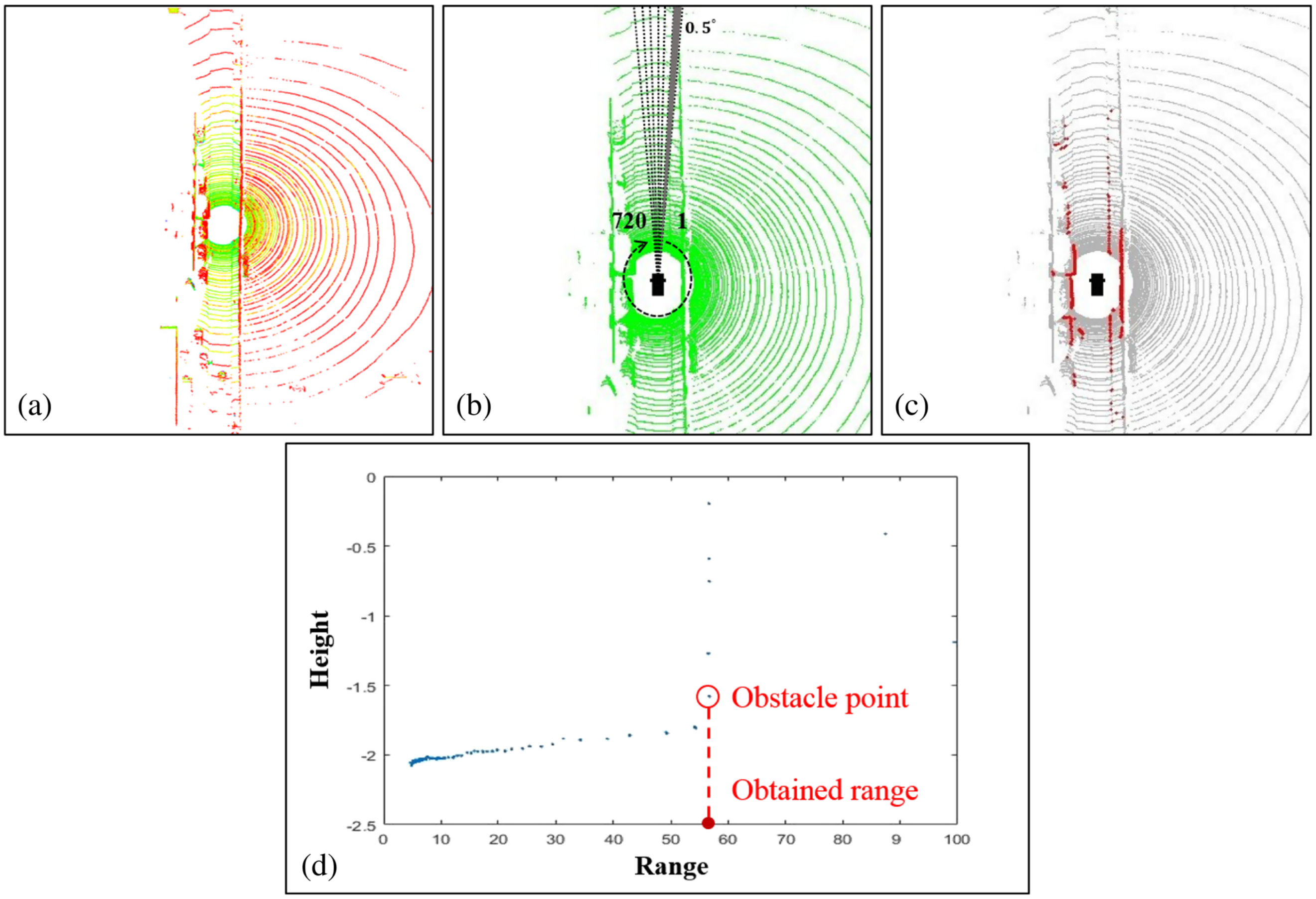
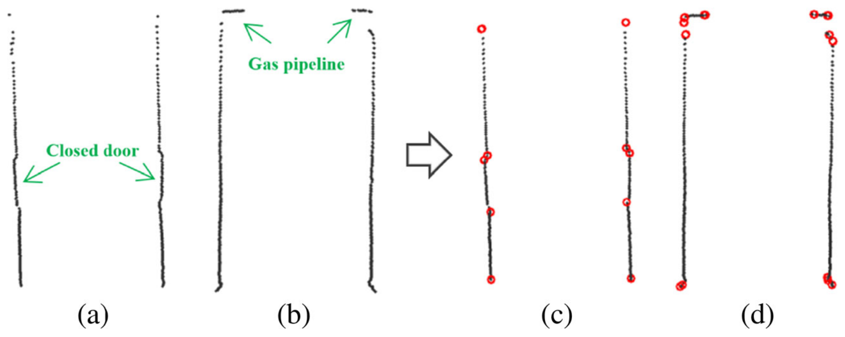


Fig. 6 720 2D obstacle scan points generation from the 3D lidar points cloud

As we introduced, dominant points are selected as the critical rays in our SLAM method. Plenty of researchers have developed great methods for finding dominant points in the fields of computational geometry and computer graphics, which has been studied, e.g. in [3, 5, 20] and so on. In the field of robotics, many researchers try to select dominant points for conducting points to points matching and points to map matching to decrease the computational complexity, such as [26] and [25]. In contrast, we don't throw away the uncritical points, but increase the importance of the critical points in Scan Matching module to enhance structural features without considering the time complexity. In our method, we adopted and improved the dominant points selection method in [3] to select the critical rays from the current scan. Since the method proposed in [3] was mainly designed for the theoretical problem and only tested for some artificial shapes, we have to improve it according to our own requirements if we want to employ it in the real system. On the one hand, considering that the distance between two scan points may become very far or very near with the change of the scanning environment, which is quiet uncertain, we use the point's serial number to set

Fig. 7 Critical rays or dominant points selection in the experiments with few structural characteristics(for example, the long-straight corridor)



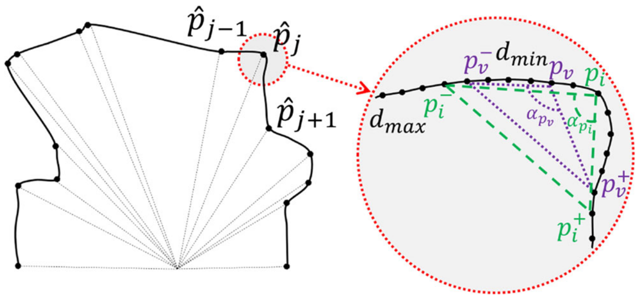


Fig. 8 Dominant points selection method from the obstacle scan points in our method

searching scope thresholds instead of the real distance in [3]. On the other hand, the computational load for serial numbers is much smaller than that for distance. In detail, as shown in Fig. 8, in the vehicle-based coordinates system, the coordinate of the i^{th} scan point p_i is (x_i, y_i), the j^{th} selected dominant point(critical ray) is $\hat{p}_j$. Three steps are included in the modified method:

1. Calculating the curvature α_{p_i} of each scan point p_i.
2. Judging whether p_i is a dominant point candidate through comparing with its neighboring points.
3. Picking out the real dominant points from the candidates according to the proper threshold.

In the first step, for the i^{th} scan point p_i, the variable triangle (p_i^-, p_i, p_i^+) is structured to calculate several curvature candidates $\alpha_{p_i}^k$ for p_i, constrained by a set of simple rules that are shown in formula (4–6), where $I(p_i)$ is the serial number of p_i, namely $I(p_i) = i$. d_{min} and d_{max} are two parameters to determine the selection range of p_i^- and p_i^+. Thus, we can get N curvature candidates for p_i to acquire its final curvature value α_{p_i} according to formula (7).

$$d_{min} \le \left|I(p_i) - I(p_i^-)\right| \le d_{max} \tag{4}$$

$$d_{min} \le \left|I(p_i^+) - I(p_i)\right| \le d_{max} \tag{5}$$

$$\alpha_{p_i}^k = arccos \frac{\left|p_i - p_i^-\right|^2 + \left|p_i^+ - p_i\right|^2 - \left|p_i^+ - p_i^-\right|^2}{2\left|p_i - p_i^-\right|\left|p_i^+ - p_i\right|} \tag{6}$$

$$\alpha_{p_i} = \min_{1 \le k \le N} \{\alpha_{p_i}^k\}, \quad N = (d_{max} - d_{min})^2 \tag{7}$$

In the second step, p_i should be excluded from the valid candidates if it has a sharper neighbor p_v: $\alpha_{p_i} > \alpha_{p_v}(|i - v| \le d_{max})$. In this way, the final dominant point candidates are obtained. In the third step, the j^{th} dominant point $\hat{p}_j$ can be acquired if the curvature value α_{p_i} of its corresponding candidate p_i is less than the default threshold: $\alpha_{p_i} < \Phi$. In our method, $\Phi = 125°, d_{max} = 8, d_{min} = 4$.

We have compared this critical ray selection method with that in CRSM-SLAM system[26] by using its open source in different environments, which is shown in Fig. 9. In this figure, (a) and (b) show the critical rays selection result of CRSM-SLAM system in two kinds of environment, and our selection results are presented in (c) and (d) under the same condition. It is obvious that our method is able to obtain better critical rays than theirs to express the environment structure characteristics so that Scan Matching becomes more easy and accurate.

4.3 Scan Matching

In CRSPF-SLAM, an effective points to map scan matching method based on particle filtering is proposed as shown in the algorithm box. To make the flow and explanation

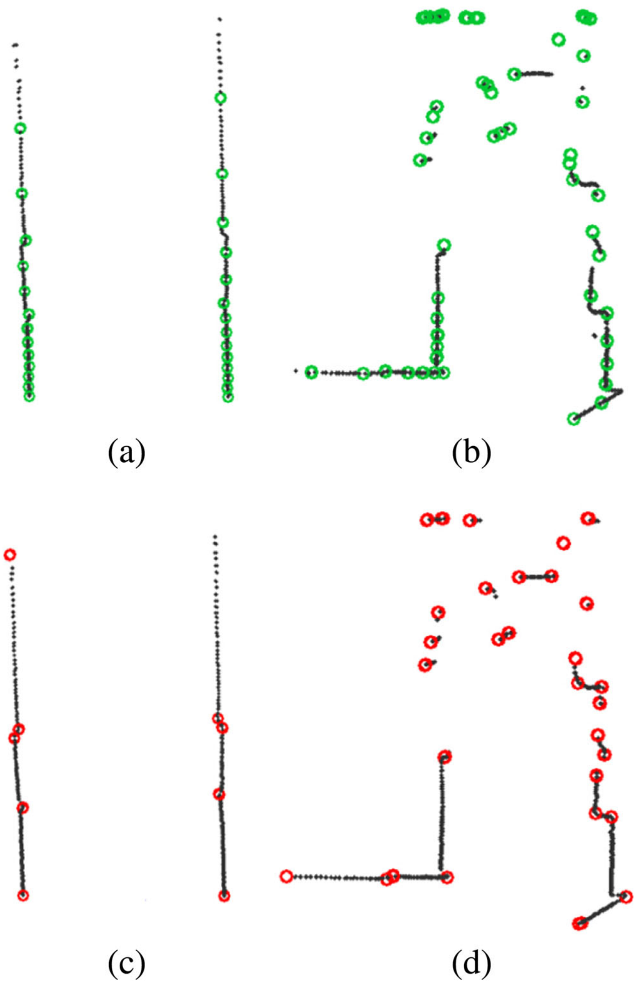


Fig. 9 Critical rays selection results comparison between our method and CRSM-SLAM in different environments

understood clearly, we present the algorithm in the form of pseudo code. In which, the basic idea of this module is converging to an "optimal" particle as the final robot pose with the corresponding scan from all the particles that are repeatedly scattered around the current observed robot pose in the map according to the dynamic filtering (or searching) range (X^M, Y^M, A^M) and the initial particles number *loopnum*, as shown in Fig. 10a–c.

In detail, the inputs of this module include the current scan $S_b(t)$ (in the robot base-link coordinate system), the current observed robot pose $\tilde{P}(t)$ (in the map coordinate system) from the odometer and the obtained match value in the last frame $M(t-1)$. Then, we can get an "optimal" robot pose $P(t)$ (in the map coordinate system) with its corresponding match value $M(t)$ for each frame. What needs to be explained is that each particle ($\{P_r(t), S_m(t)\}$) used in our method includes a random robot pose $P_r(t)$ and its corresponding scan $S_m(t)$ in the map coordinate system, which is obtained through transferring the scan $S_b(t)$ in the base-link coordinate system according to formula (8). Specifically, the category correspondence of the words presented in the algorithm box is:

- $P_r(t) : bestpose, lastbestpose, posetemp$;
- $S_m(t) : observescan, scantemp$;
- $M(t) : matchbest, lastmatchbest, matchtemp$;

In the core function, as the first step (lines 7–10), if it is the first-time system launch, all the states and parameters are initialized. Otherwise, for each frame, a feedback is designed to make scan matching method more robust, the initial particles number of each frame *loopnum* and the filtering range (X^M, Y^M, A^M) are reset according to the last match value $M(t-1)$ by using a inverse proportional function $f(x) = b\big/(x-a)$, which is shown in formula (9). Then, on the one hand, if the current scan points number $N_s(t)$ is not enough ($N_s(t) \leq \hat{N_s}$), the observed pose $\tilde{P}(t)$ with its scan is set to be the best particle and $M(t)$ is set to be 0 (line 40). In our system, the matching number threshold $\hat{N_s}$ is set to be 60 according to our empirical intuition when the scan points maximum number N_s^{max} is 720. On the other hand, when there are enough scan points in the current frame, the "optimal" particle $\{P(t), S_m(t)\}$ with the highest match value $\tilde{m}(t)$ can be obtained through an iterative convergence process(Monte Carlo), as shown in the box (lines 12–37).

$$
\begin{aligned}
S_m(t) &= \begin{bmatrix} cos\Phi(t) & -sin\Phi(t) \\ sin\Phi(t) & cos\Phi(t) \end{bmatrix} S_b(t) + \begin{bmatrix} X(t) & \dots & X(t) \\ Y(t) & \dots & Y(t) \end{bmatrix} \\
S_m(t) &= [p_m(1)\dots p_m(i)\dots p_m(N_s(t))] \\
S_b(t) &= [p_b(1)\dots p_b(i)\dots p_b(N_s(t))] \\
p_m(i) &= [x_m(i) \quad y_m(i)]^T \\
p_b(i) &= [x_b(i) \quad y_b(i)]^T
\end{aligned}
\tag{8}
$$

Algorithm Scan matching in CRSPF-SLAM

```
Input:
 1: The current scan in base-link coordinate system, $S_b(t)$;
 2: The current observed robot pose in the map coordinate system, $\tilde{P}(t) = \left\{\tilde{X}(t), \tilde{Y}(t), \tilde{\Phi}(t)\right\}$;
 3: The obtained match value in last frame, $M(t-1)$;
Output:
 4: The obtained "optimal" robot pose in the map coordinate system, $P(t) = \{X(t), Y(t), \Phi(t)\}$;
 5: The obtained match value in current frame, $M(t)$;

            Core function of scan matching
 6: function ScanMatch($S_b(t)$, $\tilde{P}(t)$, $M(t-1)$)
 7:     if $FIRST\_TIME$ then
 8:         Conduct initialization and empty $FIRST\_TIME$
 9:     end if
10:     Reset $loopnum$, $X^M$, $Y^M$, $A^M$ according to $M(t-1)$ through formula (9)
11:     if $N_s(t) > \hat{N_s}$ then
12:         $observescan$ = BaseScan2Map($S_b(t)$, $\tilde{P}(t)$)
13:         $matchbest$ = MatchvalCal($observescan$)
14:         $lastmatchbest = matchbest$
15:         $lastbestpose = bestpose = \tilde{P}(t)$
16:         $discountpose = discountangle = 1.0$
17:         while $counter < loopnum$ do
18:             $posetemp$ = GetRanPose($lastbestpose$)
19:             $scantemp$ = BaseScan2Map($S_b(t)$, $posetemp$)
20:             $matchtemp$ = MatchvalCal($scantemp$)
21:             if $matchtemp > matchbest$ then
22:                 $matchbest = matchtemp$
23:                 $bestpose = posetemp$
24:             else
25:                 $counter$++
26:             end if
27:             if $counter > loopnum\big/3$ then
28:                 if $matchbest > lastmatchbest$ then
29:                     $counter = 0$
30:                     $discountpose = discountpose * 0.3$
31:                     $discountangle = discountangle * 0.2$
32:                     $lastbestpose = bestpose$
33:                     $lastmatchbest = matchbest$
34:                 end if
35:             end if
36:         end while
37:         $P(t) = bestpose, \tilde{m}(t) = matchbest$
38:         Caculate $M(t)$ according to formula (11)
```

```
39:     else
40:         $P(t) = \tilde{P}(t)$, $M(t) = 0$
41:     end if
42:     $M(t-1) = M(t)$
43: end function

            Other functions introduction in brief
44: function BASESCAN2MAP($S_b(t)$, $robotpose$)
45:     Obtain the current scan $S_m(t)$ in the map coordinate
        system through coordinates transformation according to
        the robot pose in the map based on formula (8).
46:     return $S_m(t)$
47: end function
48: function MATCHVALCAL($curscan$)
49:     Calculate the match value $m(t)$ by matching current
        scan with the map according to formula (10).
50:     return $m(t)$
51: end function
52: function GETRANPOSE($robotpose$)
53:     Get a random pose $P_r(t)$ around the current robot
        pose:
            $X_r(t) = ran(-X^M, X^M) * discountpose$
            $Y_r(t) = ran(-Y^M, Y^M) * discountpose$
            $\Phi_r(t) = ran(-A^M, A^M) * discountangle$
54:     return $P_r(t)$
55: end function
```

$$\begin{aligned} loopnum &= \frac{(M_{max} - M_{min})n_{max}n_{min}}{(n_{max} - n_{min})M(t-1) - (M_{min}n_{max} - M_{max}n_{min})} \\ X^M &= \frac{(M_{max} - M_{min})X^M_{max}X^M_{min}}{(X^M_{max} - X^M_{min})M(t-1) - (M_{min}X^M_{max} - M_{max}X^M_{min})} \\ Y^M &= \frac{(M_{max} - M_{min})Y^M_{max}Y^M_{min}}{(Y^M_{max} - Y^M_{min})M(t-1) - (M_{min}Y^M_{max} - M_{max}Y^M_{min})} \\ A^M &= \frac{(M_{max} - M_{min})A^M_{max}A^M_{min}}{(A^M_{max} - A^M_{min})M(t-1) - (M_{min}A^M_{max} - M_{max}A^M_{min})} \end{aligned} \tag{9}$$

Specifically, in the iterative convergence process, we firstly get the observed pose with scan from the odometer module and initialize the iteration parameters for Monte Carlo, as shown in lines 12–16. Where, the two key parameters, *discountpose* and *discountangle*, are very important for the convergence process. They are always diminishing (lines 30–31) to reduce the random searching range (line 53) when the local convergence condition is met. Thus, the particle filter process can eventually converge to an "optimal" particle. In our system, the decay factors for position and angle are set to be 0.3 and 0.2, respectively. They are determined finally through repeating test to balance the convergence time and the convergence accuracy. The core part of the iterative convergence process is conducted in a loop (lines 17–36), where the basic logic of Monte Carlo is presented in detail. In this loop, besides reducing searching range and iterating the states, there are mainly three repeated functions: GETRANPOSE, BASESCAN2MAP and MATCHVALCAL. They are also explained in the algorithm box. GETRANPOSE is mainly used for generating the random particles within a certain range, in which the initial range (X^M, Y^M, A^M) for each frame is updated dynamically as we introduced above. BASESCAN2MAP is only for coordinates transformation, as shown in formula (8). It is obvious that the most important one among them is MATCHVALCAL, namely how to calculate the match value for each particle.

For the match value calculation process, a buffer zone for each particle is designed, as shown in Fig. 10d–e, to make the iteration process (the loop in the algorithm box) easier to converge so that the system becomes more robust. In practice, we have tried two ways to obtain the buffer zone with a width of w_i. Firstly, for each scan point p_i, a block with the width of w_i is designed as shown in Fig. 10f. For the first way, only the pixels around p_i at the specific direction are selected to calculate the match value. For example, $p_i^{2a}, p_i^{16}, p_i^{15}, p_i, p_i^{11}, p_i^{12}, p_i^{22}$ are the matched pixels at the direction of $\overrightarrow{Op_i}$. However, the calculated amount will increase greatly if the direction is calculated for all the scan points of each particle. In order to guarantee the real-time performance, we adopted a simple way that all the pixels in the block of p_i are used. In the experiments, we found there is little difference between the match results of these two approaches.

$$m(t) = \frac{\sum_{i=1}^{N_s(t)} \left\{ \alpha_i \left[c(p_m(i)) + \sum_{h,k=1}^{H_i,K_i} \left(\delta_k c(p_m^{kh}(i)) \right) \right] \right\}}{\sum_{i=1}^{N_s(t)} \alpha_i} \tag{10}$$

$$M(t) = \tilde{m}(t) + \frac{\max\{0, (N_s(t) - N_s(t-1))\}}{N_s^{max}} \tilde{m}(t-1) \tag{11}$$

Based on this matching model, the match value for each particle $m(t)$ is calculated according to formula (10). In this formula, $c(p_m(i))$ is the pixel value of the i^{th} scan point $p_m(i)$ (in the m^{th} particle) in the map. Similarly, $c(p_m^{kh}(i))$ is the pixel value of the kh^{th} buffer zone pixel of $p_m(i)$. And δ_k is the attenuation coefficient(less than 1.0) of the buffer zone, which decreases with k increases. For example,it is set to be $\delta_k = \left\{\frac{2}{3}, \frac{1}{3}\right\}$ when $k = \{1, 2\}$ in our system, respectively. As an empirical choice, the buffer width K_i is uniformly set to be 2 in our system. What's more, we paid more attention to the selected critical rays. In order to increase the structural features (especially for the less structural feature environments, like long straight

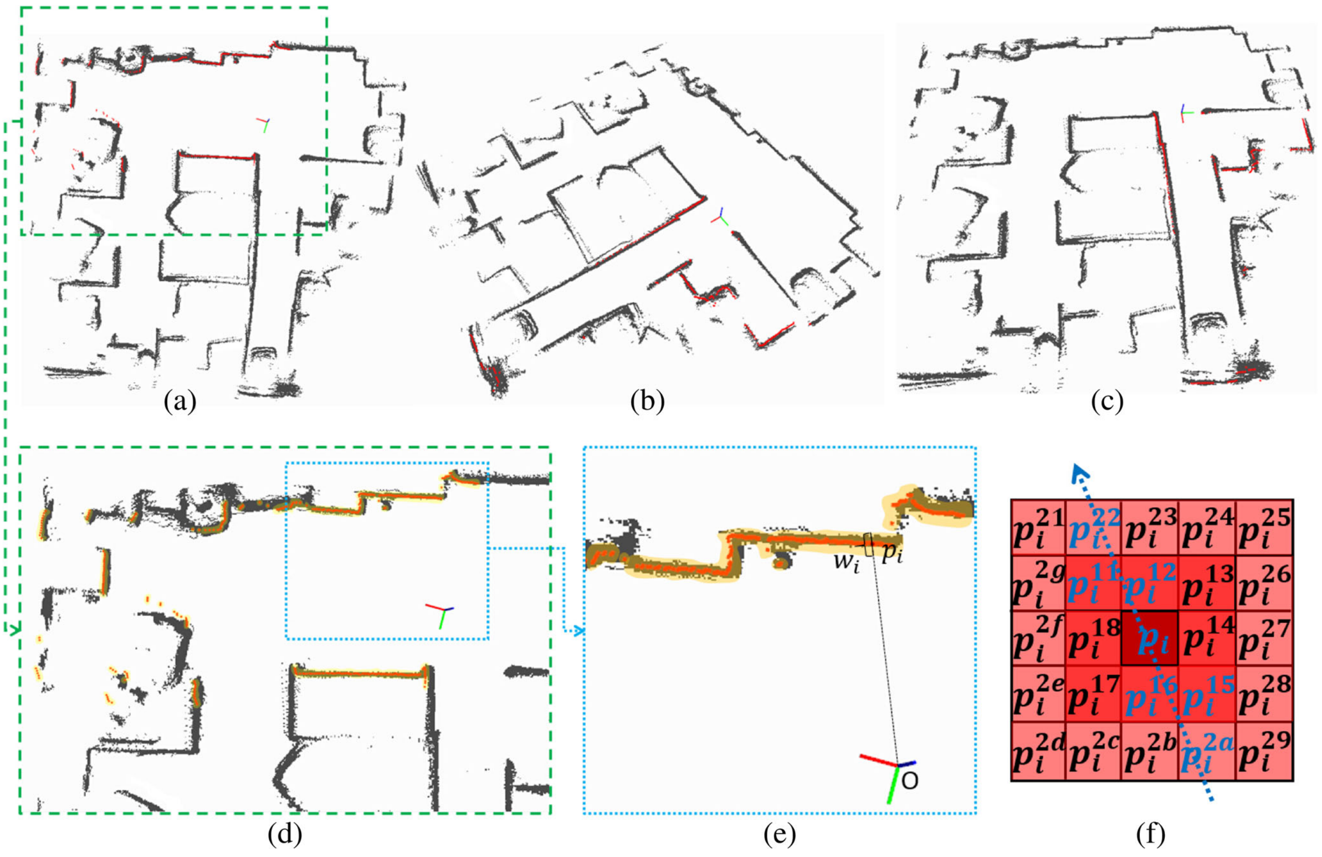


Fig. 10 Scan matching process in our method

corridor) without losing the influence of other common rays, the weight coefficient (α_i) for these two type points must be suitable. It should be set to be appropriately larger for the critical rays than that for the common rays. Through experimental testing in different environments, we set α_i to be 10 if $p_m(i)$ is the critical ray. Otherwise, α_i is set to be 1. In this way, the match value for each particle $m(t)$ can be calculated objectively and the best match value $\tilde{m}(t)$ for the "optimal" particle can be obtained after the iterative convergence process.

Finally, a filtered match value of this frame $M(t)$ is calculated as the feedback for the next frame according to formula (11). In which, the best match value of the last frame $\tilde{m}(t-1)$ is taken into account. When the number of scan points $N_s(t)$ increases suddenly, the influence of $\tilde{m}(t-1)$ will be strengthened. For example, $\tilde{m}(t)$ may decrease if a new wall is scanned for the first time according to formula (10), but $M(t)$ won't decrease if the localization in the last frame is accurate. Correspondingly, if the scan points number doesn't change greatly between two adjacent frames, $M(t)$ is approximately equal to $\tilde{m}(t)$. Thus, $M(t)$ can be used to characterize the current positioning accuracy of the robot.

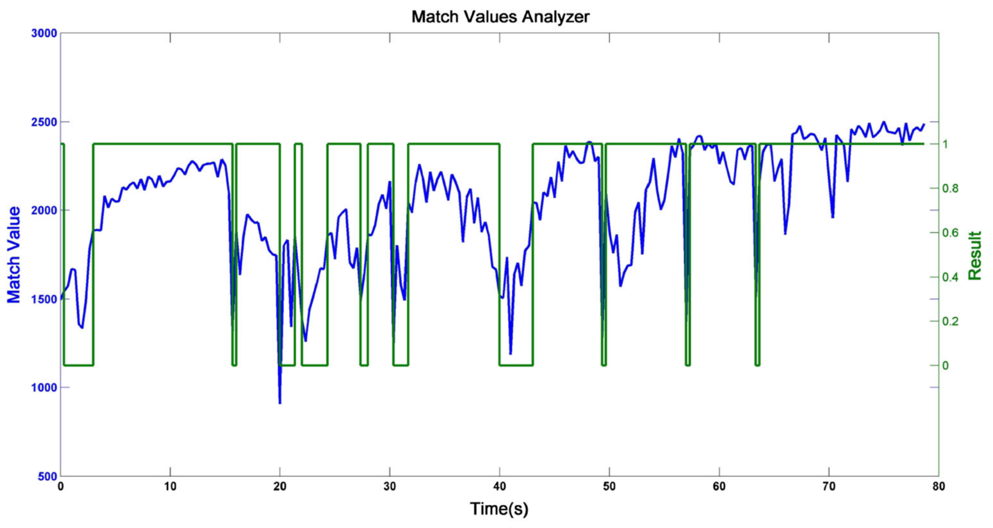
4.4 Match Values Analyzer

In CRSPF-SLAM system, match values analyzer is another extremely important part, that decides whether the map can be updated by using the obtained particle and whether the "optimal" pose can be given back to the Odometry module for eliminating its cumulative error.

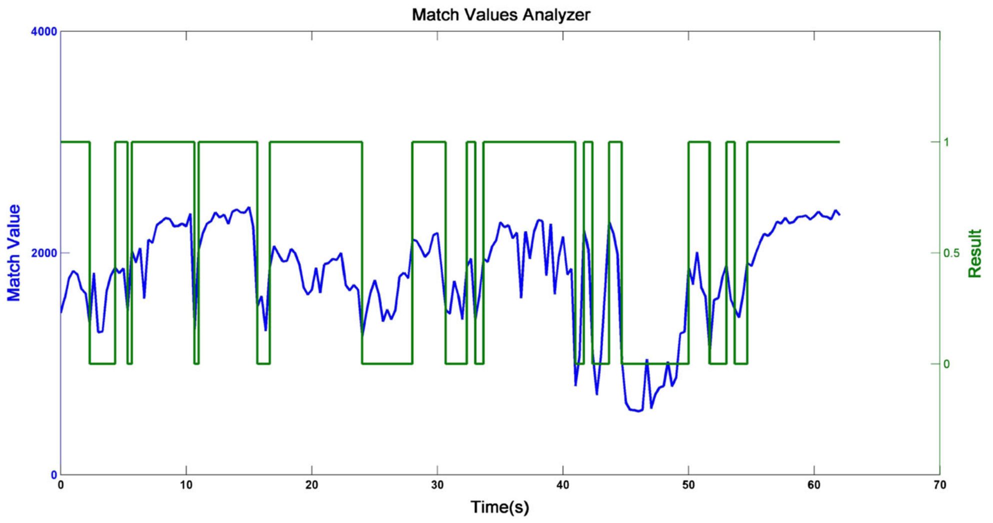
$$Result(t) = \begin{cases} \begin{cases} = 1 & M(t) > M_u \\ = 0 & else \end{cases} & , Result(t-1) = 0 \\ \begin{cases} = 0 & M(t) < M_l \\ = 1 & else \end{cases} & , Result(t-1) = 1 \end{cases} \tag{12}$$

A hysteresis comparator is employed in our method to get the analysis result that is shown in formula (12), where, the upper threshold M_u is set to be 1850 and the lower threshold M_l is set to be 1550 in our system. In Fig. 11, the match value (blue line) changes with robot's moving in three cases and the their results (green line) are generated, respectively. In general, $M(t)$ decreases when the robot makes a turn or moves suddenly, works in the bumpy environments and

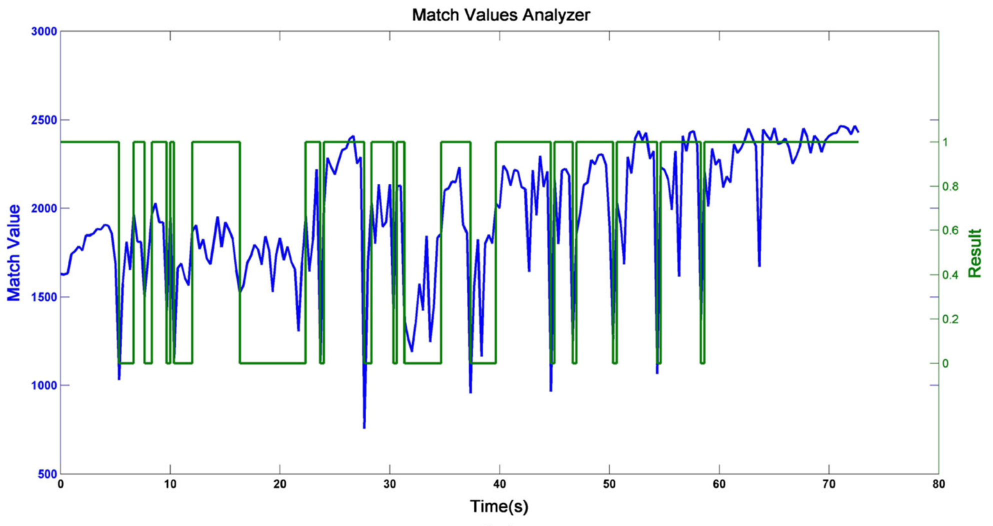
Fig. 11 Match values analyzer works in different situations



(a)



(b)



(c)

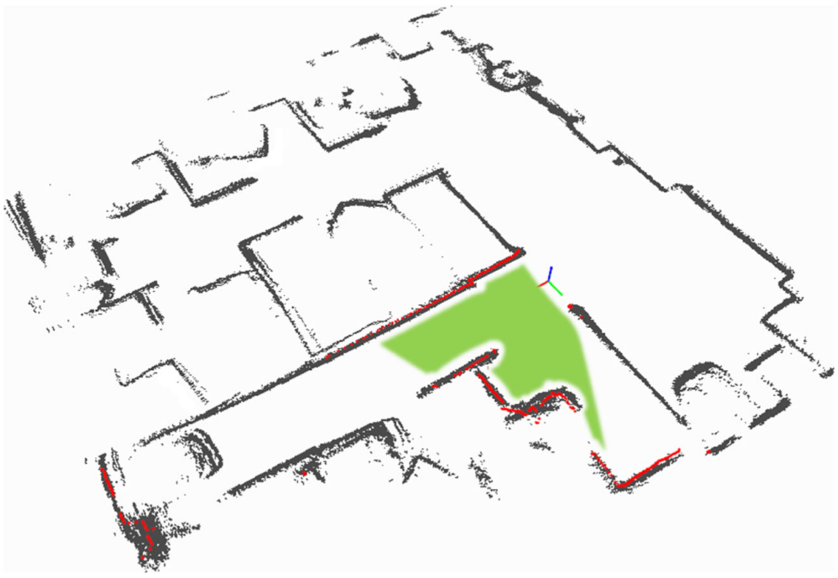


Fig. 12 Map Update works for error elimination

other disturbance occurs. Then, location of the robot can be determined to be accurate as the key frame when $Result = 1$ so that most of the noise can be eliminated.

5 Map Update

As we introduced in Section 2, Map Update module includes three process actions: draw scan in the map, remove the errors of the map and remain unchanged. Different actions are conducted according to the analysis result of Match Values Analyzer.

When $Result = 0$ in Match Values Analyzer, the localization of robot in the map is determined to be not good enough for error elimination. In this case, another hysteresis comparator is used in our system as shown in formula (13). Where, the upper threshold M_u^d is set to be 1250 and the lower threshold M_l^d is set to be 850 in our system. On the one hand, the current scan will be draw in the map if the current match value M(t) is not too bad(that is

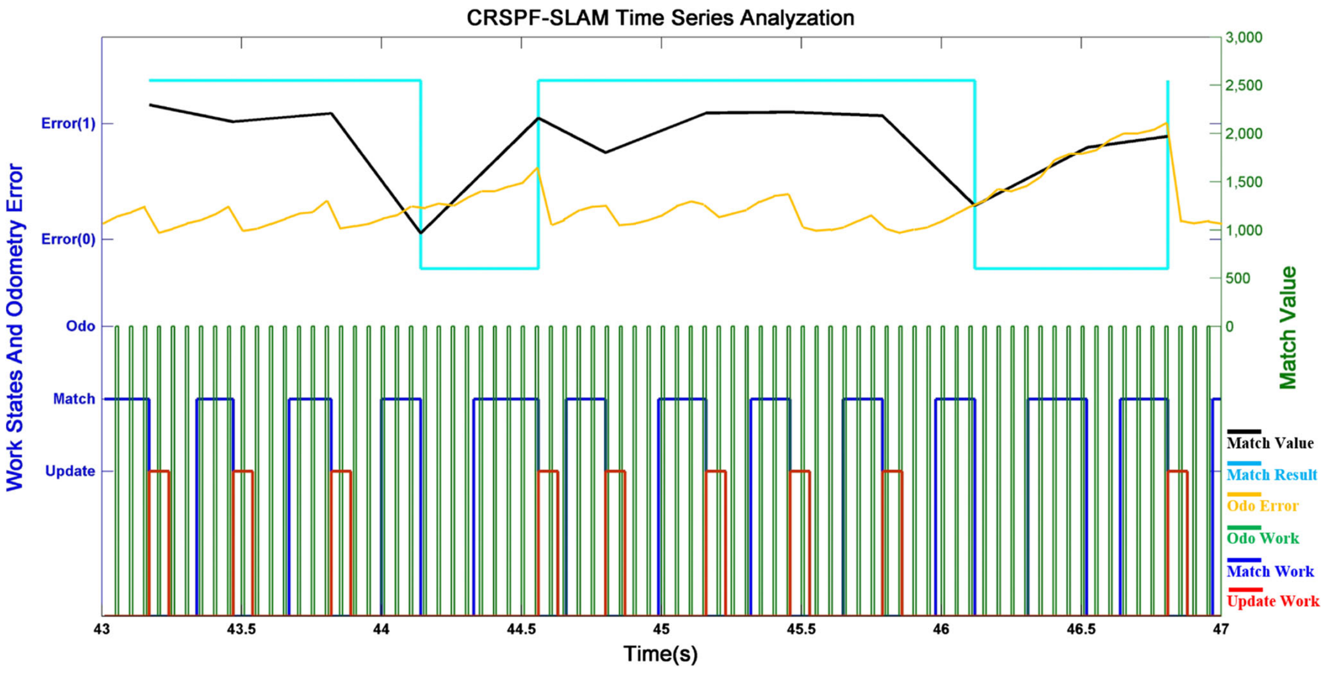


Fig. 13 Time Series Analyzation for the modules in our system

Table 1 Error estimation for different algorithm in various simulation environments

Simulation experiments

Environment	HectorSLAM	Gmapping	KartoSLAM	CoreSLAM	LagoSLAM	CRSPF-SLAM
1r5map	7.4581	5.3670	5.4380	171.5218	9.3041	6.1675
MRL Arena	0.4563	0.4200	0.5509	11.8393	1.4646	0.5310

$Draw = 1$). On the other hand, the map will not be updated (remain unchanged) with the location update of the robot if $Draw = 0$. Thus, plenty of error scans are prevented from being drawn in the map when the robot moves hastily. In other words, only not bad scans can be regarded as the key frames candidates.

$$Draw(t) = \begin{cases} \begin{cases} = 1 & M(t) > M_u^d \\ = 0 & else \end{cases} & , Draw(t-1) = 0 \\ \begin{cases} = 0 & M(t) < M_l^d \\ = 1 & else \end{cases} & , Draw(t-1) = 1 \end{cases} \tag{13}$$

When $Result = 1$ in Match Values Analyzer, the current scan can be determined as good enough to be key frames. The key frames are not only used to draw in the map, but also used for error elimination. Specifically, the near and stable scan points of the obtained key frame are selected to get a passable area. Then, the passable area is corroded to be a smaller area (green area in Fig. 12) for error elimination. In our system (iRobot, for example), only the scan points within 3 m are selected to generate this area when the detection range of lidar is 5 m considering that the far scan points are not stable enough. In this way, the previous error drawing and the trace caused by dynamic obstacles can be erased accurately.

6 Time Series Analyzation

As we stated above, all the modules of our system are designed to work asynchronously. In order to explain the relationship between them clearly, we analyzed the time series of our system as shown in Fig. 13.

In detail, the odometry module is set to work at the rate of $10Hz$ (green histogram shown in Fig. 13) and the cost time of it is almost constant. The particle filtering module is set to work at the rate of $3Hz$ (blue histogram shown in Fig. 13) and its operation time is variable according to the last match value. And the map update module is set to work or not according to the analysis result of the match values analyzer as the red histogram shown in Fig. 13. Specifically, this module works only when the match result (light blue line in Fig. 13) becomes 1. And the match result is determinated according to the match values (black line in Fig. 13) by using a hysteresis comparator. Meanwhile, to prove the effectiveness of the algorithm, the odometer error (yellow line) is also shown in Fig. 13. It is obvious that the cumulative error of the odometer can be eliminated through the low-frequency scan matching and map updating so that the localization and mapping system can keep stable.

7 Experimental Results

In order to verify the effectiveness of our method and compare it with other existing 2D SLAM methods, both stage simulations and plenty of real world experiments on

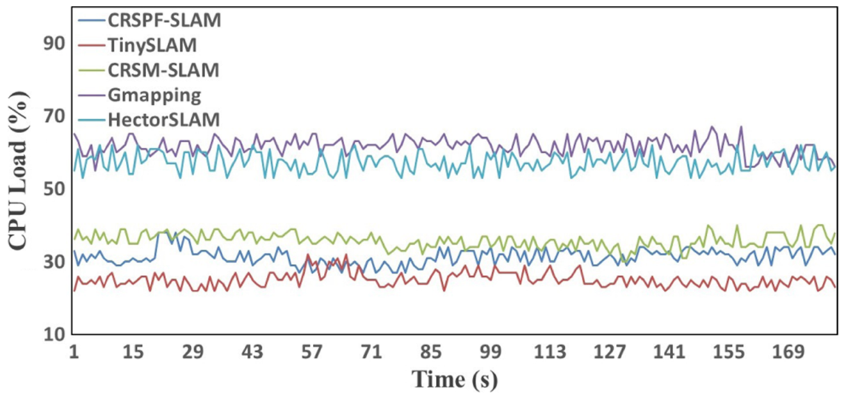


Fig. 14 Evolution of the CPU load of each SLAM method under the same condition

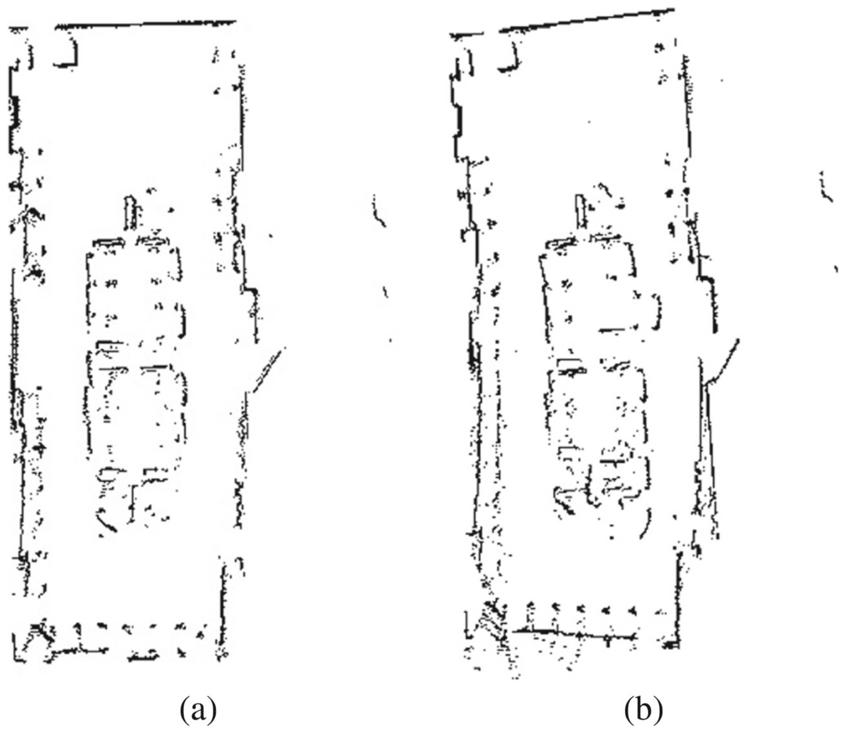


Fig. 15 Mapping results comparison in a large meeting room between our method (**a**) and HectorSLAM (**b**)

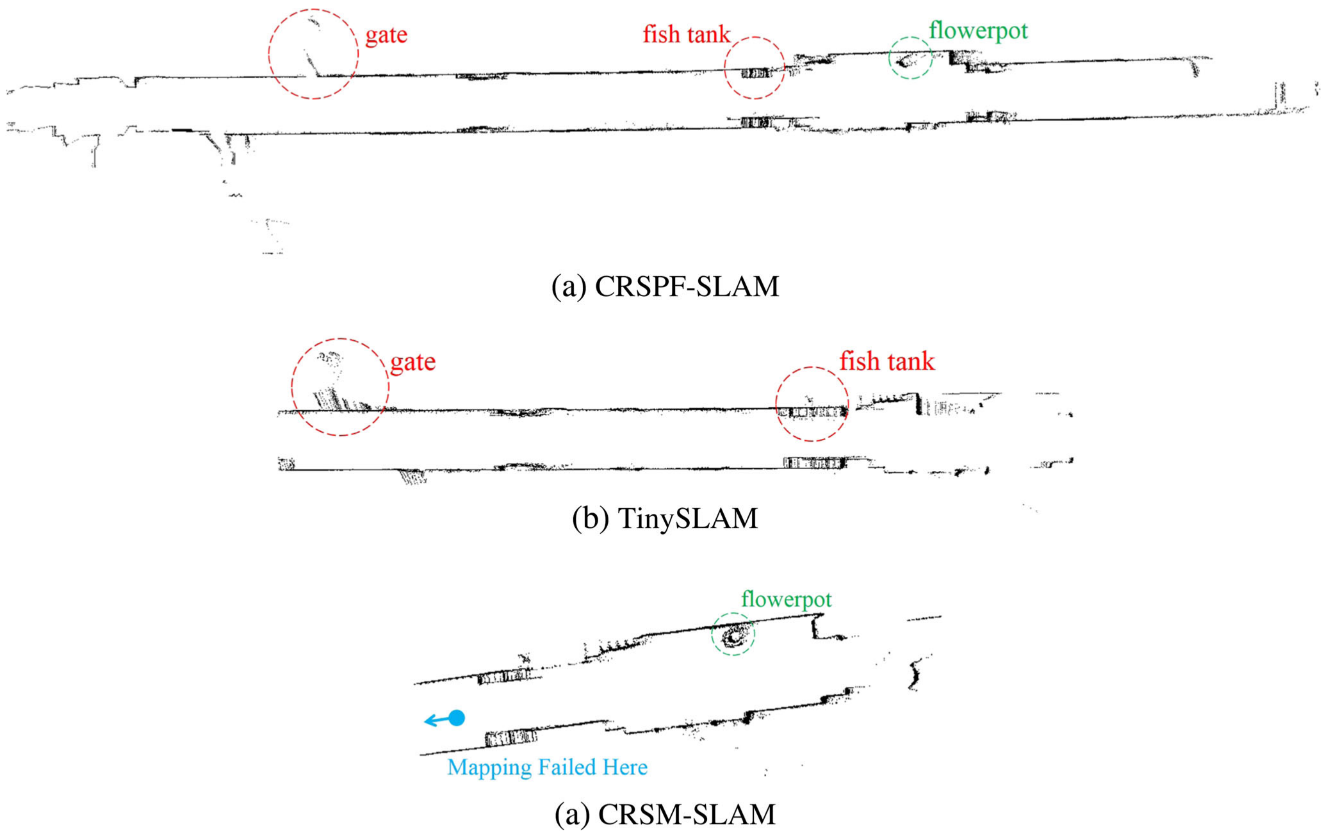
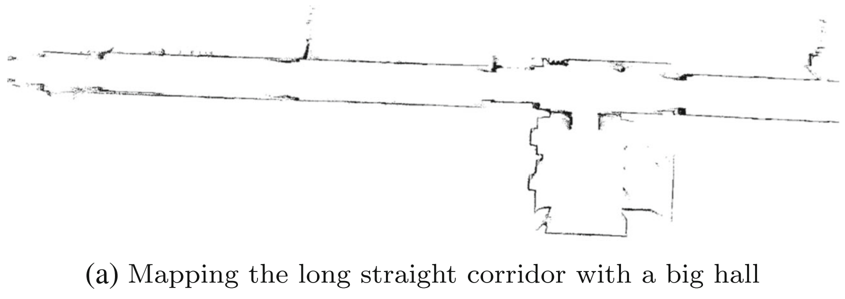


Fig. 16 Experimental results comparison in the long straight corridor which is as long as 80 m between our mthod, TinySLAM and CRSM-SLAM



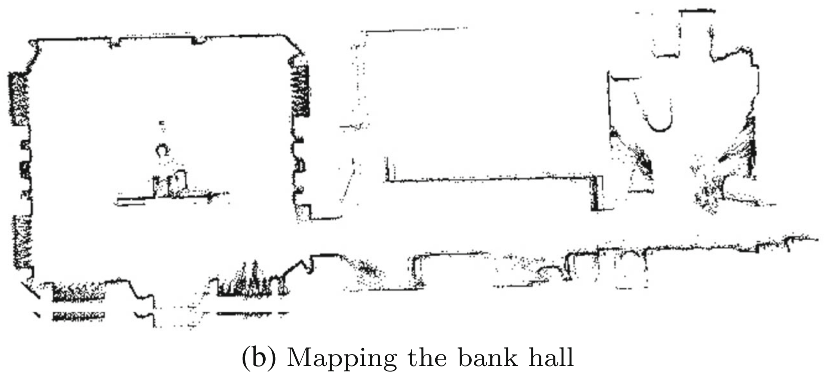


Fig. 17 Indoor experiments for our method

(a) Mapping the long straight corridor with a big hall

(b) Mapping the bank hall

different kinds of platforms were performed. On the one hand, considering that it's so expensive and difficult to get the ground truth data for the accurate errors calculation, we mainly evaluate and compare the errors or accuracy of these methods quantificationally through simulations. On the other hand, we also present and compare the real world experimental results of the different methods qualitatively.

For the simulation experiments, we mainly employed two different maps: MRL Arena and 1r5map, which were introduced detailedly in [23]. Their paper detailedly evaluated and compared the common 2D SLAM approaches by providing the specific errors and parameters based on their designed evaluation method. The approaches presented in their paper include HectorSLAM, Gmapping, KartoSLAM, CoreSLAM and LagoSLAM. What's more, the methods and their source code link were also introduced and presented in their paper. Specifically, in [23], the authors conducted the simulation experiments by using the toll – Stage (MRL Arena and 1r5map can be found in this toll), which is a realistic 2D robot simulator integrated in ROS. They employed Hokuyo URG-04LX-UG01 and set same parameters for all the methods in the simulation environment. What's more, they conducted an analysis of the error between the generated map and the ground truth to evaluate the quality of the obtained maps. Specifically, they used a performance metric based on the k-nearest neighbor concept, that is computing the best fit alignment between the ground truth and the map by using intensity-based image registration tools. The error calculation process was introduced detailedly in their paper. We conduct the simulation experiments based on their evaluation system and test results. To compare with their test results for different methods, the simulation parameters and computational condition are set same (or very close) as theirs. In this way, we obtained the map errors of our method for these two environments (MRL Arena and 1r5map), which are shown in Table 1. According to the simulation experimental results, we can find our method can get the similar map errors with the valuable methods, like Gmapping, in these two environments. However, as

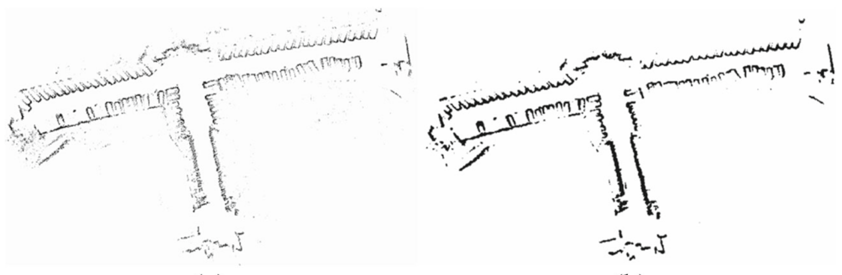


Fig. 18 Filtering and one-time expansion for the outdoor system

(a) (b)

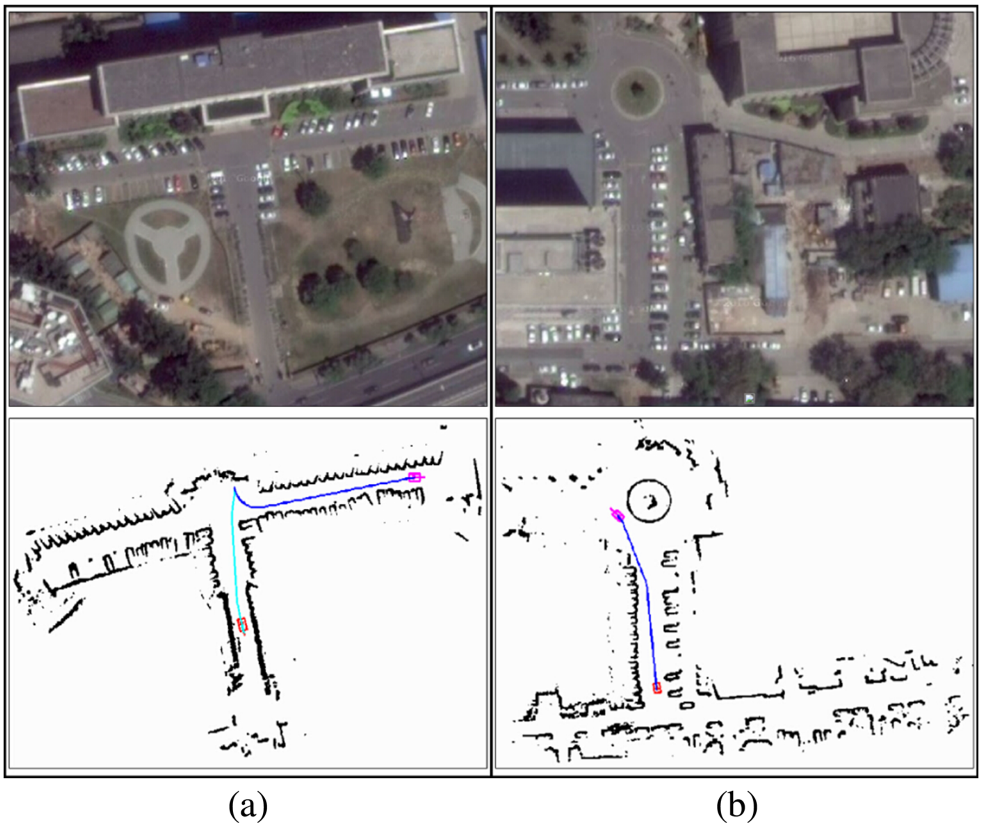


Fig. 19 Outdoor experiments for our method

we explained above, our method is a lightweight SLAM method, which requires lower calculated cost than other enormous methods, which can be proved in Fig. 14. As we all know, SLAM methods perform quite differently when the processor performance is different. For example, it is usually very hard for the complex methods to get good results when they works on the low performance computer, especially with the changing of working environment and working time. Therefore, we also conducted some real-world comparison experiments with the enormous method and other lightweight methods (low computational cost methods shown in Fig. 14).

For the real-world experiments, we tried to use three kinds of odometers in our system for different platforms, respectively. As shown in Fig. 15, we compared our method with HectorSLAM in a large meeting room whose size is about 30 m×10 m. In this experiment, we employed the small platform iRobot and combined its wheel odometer (Fig. 3) with scan matching to conduct SLAM. The experimental results show that our method is much easier than HectorSLAM to get an accurate map under the same conditions. The system on which the algorithms were executed, is a mediocre notebook computer with Intel Core i3-5005U CPU, 4 GB RAM, running Ubuntu 14.04.

For the lidar or radar SLAM system, the long straight experiments are extremely challenging because of the few architectural features at the longitudinal direction. Therefore, we conducted some experiments in this kind of environment to test the performance of different methods. For example, Fig. 16 shows the experimental results for a long straight corridor which is as long as 80 m without enough architectural features. Specifically, Gmapping, TinySLAM and CRSM-SLAM were compared with our method on the iRobot platform whose detection range is 5 m in the experiment. The traveling speed of the robot was set to be about 1.5 m/s, and its rotate speed was set to be about 45°/s. Then we found that the real-time capability of Gmapping[12] was unable to meet the requirement of the system because it's not lightweight enough. Besides, as shown in Fig. 16b, it's obvious that the long straight corridor map created by TinySLAM is not very accurate at the longitudinal direction. For example, there are much ghosting of the gate and fish tank in its map at the longitudinal direction compared with our method (Fig. 16a). And we found CRSM-SLAM failed to update the map when the robot is moving toward the left side in the corridor as shown in Fig. 16c because there is few features for its matching.

What's more, we also conducted some experiments on other platforms to test our method. On the one hand, Fig. 17 shows the indoor experimental results of a bank robot which is made by ourselves. Its calculating system is an industrial PC with Intel Core i3-4170 CPU and 4 GB RAM. In

Fig. 17a, its wheel odometer (Fig. 3) was used to map a long straight corridor with a big hall. In Fig. 17b, the stereo visual odometer was used to map in an real bank hall. The results indicate that our method can work effectively as a service robot in different indoor environments with different types of odometer. On the other hand, plenty of outdoor experiments were also done on our different autonomous vehicles. However, we found there was always some noise in the outdoor map created by our cars because the scan obstacle points in our method were obtained after dealing with the 3D points cloud. Therefore, a simple median filtering and one-time expansion are conducted on the outdoor map in our method to make the path planning easier, which is shown in Fig. 18. And Fig. 19 shows the outdoor experimental results of two autonomous cars whose calculating system is an industrial PC with Intel Core i5-6500 CPU and 4 GB RAM. In Fig. 19a, our autonomous car employed the IMU odometer to map a part of the campus and drove by itself. In Fig. 19b, the wheel odometer (Fig. 4) was used for the other car to map a parking lot in our campus and parked itself. These outdoor experimental results prove that our method can meet the requirement of some tasks, such as autonomous parking and community transportation.

Fig. 20 Product equipped with our method: Bank Robot – ONE

8 Conclusion

A lightweight but robust SLAM method for the indoor and outdoor applications is presented in this paper. Three different odometers are tried to be used for working with a self-adaptive particle filtering algorithm, respectively. Plenty of experiments conducted on different kinds of platform in a variety of environments indicate that our method is more effective than others. What's more, our method has been used for many indoor and outdoor products successfully. For example, the bank robot named ONE as shown in Fig. 20 is the collaborative product of our laboratory and China Minsheng Bank to work as a lobby manager, which has about two hundred orders in China now. Besides, our method is also used for some cars to realize the autonomous parking function. To achieve better academic exchanges, we will share our source code and experimental videos on our website. In the future, only the stereo cameras will be used in our system instead of lidar so that the cost of our products can be lowered greatly.

References

1. Bengtsson, O., Baerveldt, A.J.: Robot localization based on scan-matching—estimating the covariance matrix for the idc algorithm. Robot. Auton. Syst. **44**(1), 29–40 (2003)
2. Blanco, J.L.: Derivation and implementation of a full 6d ekf-based solution to bearing-range slam. University of Malaga, Spain, http://babel.isa.uma.es/~jlblanco/papers/RangeBearingSLAM6D.pdf, Technical Report (2008)
3. Chetverikov, D.: A simple and efficient algorithm for detection of high curvature points in planar curves. In: International Conference on Computer Analysis of Images and Patterns, pp. 746–753. Springer (2003)
4. Chong, K.S., Kleeman, L.: Accurate odometry and error modelling for a mobile robot. In: IEEE International Conference on Robotics and Automation, 1997. Proceedings., 1997, vol. 4, pp. 2783–2788. IEEE (1997)
5. Cronin, T.M.: A boundary concavity code to support dominant point detection. Pattern Recogn. Lett. **20**(6), 617–634 (1999)
6. Del Moral, P.: Non-linear filtering: interacting particle resolution. Markov Processes and Related Fields **2**(4), 555–581 (1996)
7. Diosi, A., Kleeman, L.: Laser scan matching in polar coordinates with application to slam. In: IEEE/RSJ International Conference on Intelligent Robots and Systems, 2005.(IROS 2005). 2005, pp. 3317–3322. IEEE (2005)
8. Dryanovski, I., Valenti, R.G., Xiao, J.: Fast visual odometry and mapping from rgb-d data. In: IEEE International Conference on Robotics and Automation (ICRA), 2013, pp. 2305–2310. IEEE (2013)
9. Eliazar, A., Parr, R.: Dp-slam: fast, robust simultaneous localization and mapping without predetermined landmarks. In: IJCAI, vol. 3, pp. 1135–1142 (2003)

10. Engel, J., Schöps, T., Cremers, D.: Lsd-slam: large-scale direct monocular slam. In: Computer Vision–ECCV 2014, pp. 834–849. Springer (2014)
11. Geiger, A., Ziegler, J., Stiller, C.: Stereoscan: dense 3d reconstruction in real-time. In: Intelligent Vehicles Symposium (IV), 2011 IEEE, pp. 963–968. IEEE (2011)
12. Grisetti, G., Stachniss, C., Burgard, W.: Improving grid-based slam with rao-blackwellized particle filters by adaptive proposals and selective resampling. In: Proceedings of the 2005 IEEE International Conference on Robotics and Automation, 2005. ICRA 2005, pp. 2432–2437. IEEE (2005)
13. Gutmann, J.S.: Robuste navigation autonomer mobiler systeme. Aka (2000)
14. Kerl, C., Sturm, J., Cremers, D.: Dense visual slam for rgb-d cameras. In: IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2013, pp. 2100–2106. IEEE (2013)
15. Kohlbrecher, S., Von Stryk, O., Meyer, J., Klingauf, U.: A flexible and scalable slam system with full 3d motion estimation. In: IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR), 2011, pp. 155–160. IEEE (2011)
16. Kümmerle, R., Grisetti, G., Strasdat, H., Konolige, K., Burgard, W.: g 2 o: a general framework for graph optimization. In: IEEE International Conference on Robotics and Automation (ICRA), 2011, pp. 3607–3613. IEEE (2011)
17. Leonard, J.J., Durrant-Whyte, H.F., Cox, I.J.: Dynamic map building for an autonomous mobile robot. Int. J. Robot. Res. **11**(4), 286–298 (1992)
18. Lingemann, K., Surmann, H., Nüchter, A., Hertzberg, J.: Indoor and outdoor localization for fast mobile robots. In: IEEE/RSJ International Conference on Intelligent Robots and Systems, 2004.(IROS 2004). Proceedings. 2004, vol. 3, pp. 2185–2190. IEEE (2004)
19. Lu, F., Milios, E.: Robot pose estimation in unknown environments by matching 2d range scans. J. Intell. Robot. Syst. **18**(3), 249–275 (1997)
20. Marji, M., Siy, P.: A new algorithm for dominant points detection and polygonization of digital curves. Pattern Recogn. **36**(10), 2239–2251 (2003)
21. Mur-Artal, R., Montiel, J., Tardos, J.D.: Orb-slam: a versatile and accurate monocular slam system. IEEE Trans. Robot. **31**(5), 1147–1163 (2015)
22. Rodriguez-Losada, D., Minguez, J.: Improved data association for icp-based scan matching in noisy and dynamic environments. In: IEEE International Conference on Robotics and Automation, 2007, pp. 3161–3166. IEEE (2007)
23. Santos, J.M., Portugal, D., Rocha, R.P.: An evaluation of 2d slam techniques available in robot operating system. In: IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR), 2013, pp. 1–6. IEEE (2013)
24. Steux, B., Hamzaoui, O.E.: Tinyslam: a slam algorithm in less than 200 lines c-language program. In: 11th International Conference on Control Automation Robotics & Vision (ICARCV), 2010, pp. 1975–1979. IEEE (2010)
25. Taleghani, S., Sharbafi, M.A., Haghighat, A.T., Esmaeili, E.: Ice matching, a robust mobile robot localization with application to slam. In: 22nd IEEE International Conference on Tools with Artificial Intelligence (ICTAI), 2010, vol. 1, pp. 186–192. IEEE (2010)
26. Tsardoulias, E., Petrou, L.: Critical rays scan match slam. J. Intell. Robot. Syst. **72**(3-4), 441–462 (2013)
27. Wong, R., Xiao, J., Joseph, S.L., Shan, Z.: Data association for simultaneous localization and mapping in robotic wireless sensor networks. In: IEEE/ASME International Conference on Advanced Intelligent Mechatronics (AIM), pp. 459–464. IEEE (2010)
28. Zhu, H., Fu, M., Yang, Y., Wang, X., Wang, M.: A path planning algorithm based on fusing lane and obstacle map. In: IEEE 17Th International Conference on Intelligent Transportation Systems (ITSC), 2014, pp. 1442–1448. IEEE (2014)

Wenjie Song received the B.S. degree in automation from Beijing Institute of Technology, China, in 2013. He is currently working toward the Ph.D. degree in intelligent robots with State Key Laboratory of Intelligent Control and Decision of Complex Systems, and also the School of Automation, Beijing Institute of Technology. Since 2016, he has also been studying in Princeton University as a visiting student.

His research interests include autonomous vehicle, computer vision, visual SLAM, path planning, motion controlling and so on. And he has taken part in "China Intelligent Vehicle Future Challenge" for several times as the captain or the core member.

Yi Yang received the control science and engineering Ph.D. degree in School of Automation from Beijing Institute of Technology, Beijing, China.

He is currently associate professor in School of Automation at Beijing Institute of Technology and he joined IEEE in 2010. His research interests are environment perception of unmanned ground vehicle, motion planning and control, robot design and development. He is author/co-author of more than 40 conference and journal papers in the area of unmanned ground vehicle.

Mengyin Fu received the B.S. degree from Liaoning University, China, M.S. degree from Beijing Institute of Technology, China, and Ph.D. degree from Chinese Academy of Sciences.

He was elected as the Yangtze River scholar Distinguished Professor in 2009, and won the Guanghua Engineering Science and Technology Award for Youth Award in 2010. He has gotten National Science and Technology Progress Award for several times in recent years.

He is the president of Nanjing University of Science and Technology and his research interest covers integrated navigation, intelligent navigation, image processing, learning and recognition as well as their applications.

Alain Kornhauser is Professor of Operations Research and Financial Engineering at Princeton University, where he is also Director of the Interdepartmental Transportation Research Program, and Faculty Chair of Princeton Autonomous Vehicle Engineering (PAVE). His recent research includes the development and application of activity-based trip synthesizers, large-scale network models, stochastic optimization techniques, and computer graphics to transportation problems. He is also studying fundamental aspects of human information processing, which he is applying to the design of deep learning neural networks aimed at the development of autonomous vehicles.

Meiling Wang received the B.S. degree in automation from the Beijing Institute of Technology, China, in 1992, and the M.S. and Ph.D. degrees from Beijing Institute of Technology, China, in 1995 and 2007, respectively. She has been teaching in Beijing Institute of Technology since 1995, and worked in University of California San Diego as a visiting scholar in 2004.

She was elected as the Yangtze River scholar Distinguished Professor in 2014. And she is currently the Director of Integrated navigation and intelligent navigation laboratory, Beijing Institute of Technology, China. Her research interests include advanced technology of sensing and detecting and vehicle intelligent navigation.