Research and Implementation of Robot Path Planning Based onVSLAM

Zi-Qiang Wang[1], He-Gen Xu[2], You-Wen Wan[3]

[1]*College of Electronics and Information Engineering, Tongji University,Shanghai, China*
[2] *College of Electronics and Information Engineering, Tongji University, Shanghai, China*
[3] *College of Electronics and Information Engineering, Tongji University, Shanghai, China*

Abstract. In order to solve the problem of warehouse logistics robots planpath in different scenes, this paper proposes a method based on visual simultaneous localization and mapping (VSLAM) to build grid map of different scenes and use A* algorithm to plan path on the grid map. Firstly, we use VSLAMto reconstruct the environment in three-dimensionally. Secondly, based on the three-dimensional environment data, we calculate the accessibility of each grid to prepare occupied grid map (OGM) for terrain description. Rely on the terrain information, we use the A* algorithm to solve path planning problem. We also optimize the A* algorithm and improve algorithm efficiency. Lastly, we verify the effectiveness and reliability of the proposed method by simulation and experimental results.

1 Introduction

Using robots in warehouses can greatly improve the efficiency of warehouse transfer, and the path planning of warehousing and logistics robots is the key to improving efficiency. Path planning based on occupied grid map is a common method of environment for robots, such as Dijkstra[1], A*[2], D*[3] and so on. Dijkstra's main application is to find the shortest path between the start and end point of the map, but the path search is non-heuristic and slow. D* mainly solves the problem of the change of travel cost caused by the dynamic change of the environment. For a static environment such as a warehouse, the A* algorithm can find the shortest path between two points more quickly and efficiently[4-5]. While a warehouse is a static environment, robots need the ability to rapidly map the environment if they need to be deployed quickly to different warehouses.

Monocular vision SLAM technology can make robots to build maps. There are two types of SLAM visual odometer, which are divided into direct method and feature point method [6]. Feature point method mainly includes PTAM[7], ORB-SLAM[8] and other algorithms.Direct methods are mainly LSD-SLAM [9-10], DSO [11-12] and other algorithms. The advantage of the feature point method is that the map drifts small and the loop-closures detection is accurate. However, due to the relatively sparse feature points, it can be used for finding robot localization, difficult to use for robot navigation and path planning, and relativelycalculate slow. Direct method can extract relatively dense environmental features for robot navigation and positioning, and can run in real time on a PC. But the map drift is more terrible than the feature point method, loop-closures detection time is long.

This paper mainly considers how to let the robot quickly perceive the environment and complete the path planning. First of all,direct method SLAM are used to construct the 3D point cloud map of the warehouse environment, then makeoccupied grid map according to barrier in 3D map. Secondly, this paper uses A* algorithm to plan the path in the robot's working environment.

2 Map preparation

2.1Large-scale direct monocular SLAM

Jakob Engel and Daniel Cremers at the Technical University of Munich proposed a monocular SLAM algorithm based on direct method to construct a large-scale, globally consistent environment map called Large-Scale Direct (LSD) Monocular SLAM [9-10]. Vladyslav-Usenko and Jakob Engel implemented a three-dimensional reconstruction of the real-time street environment on the moving cars[13]. Jakob Engel, JorgStückler achieved accurate depth estimation and relatively dense three-dimensional reconstruction on a stereo camera [14]. For static, stable light conditions indoor, warehouse and other environments, the use of LSD-SLAM can make the robot has a precise positioning and map building capabilities.

The algorithm consists of three major components: tracking, depth map estimation and map optimizationas visualized in Fig.1:

1. The tracking component continuously tracks new camera images. That is,it estimates their rigid body pose $\boldsymbol{\xi} \in \mathfrak{se}(3)$with respect to the currentkeyframe, using the pose of the previous frame as initialization.

2. The depth map estimation component uses tracked frames to either refineor replace the current keyframe. Depth is refined by filtering over manyper-pixel, small-baseline stereo comparisons coupled with interleaved spatialregularization as originally proposed in [9]. If the camera has moved too far,a new keyframe is initialized by projecting points from existing, close-bykeyframes into it.

3. Once a keyframe is replaced as tracking reference – and hence its depth mapwill not be refined further – it is incorporated into the global map by themap optimization component. To detect loop closures and scale-drift, asimilarity transform $\boldsymbol{\xi} \in \mathfrak{sim}(3)$ to close-by existing keyframes (includingits direct predecessor) is estimated using scale-aware, direct $\mathfrak{sim}(3)$-imagealignment.

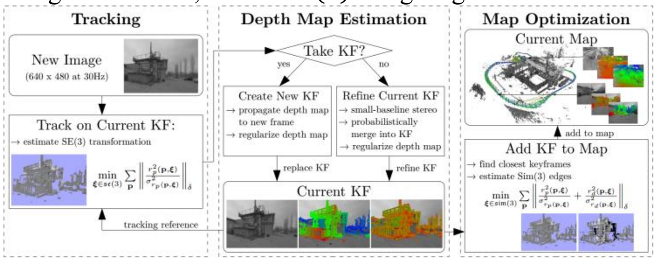
Tracking
New Image
(640 x 480 at 30Hz)
Track on Current KF:
→ estimate SE(3) transformation
$\min_{\xi \in se(3)} \sum_p \left\| \frac{r_p^2(p,\xi)}{\sigma^2_{r_p(p,\xi)}} \right\|_\delta$
tracking reference
Depth Map Estimation
Take KF?
yes
no
Create New KF
→ propagate depth map to new frame
→ regularize depth map
Refine Current KF
→ small-baseline stereo
→ probabilistically merge into KF
→ regularize depth map
replace KF
refine KF
Current KF
Map Optimization
Current Map
add to map
Add KF to Map
→ find closest keyframes
→ estimate Sim(3) edges
$\min_{\xi \in sim(3)} \sum_p \left\| \frac{r_p^2(p,\xi)}{\sigma^2_{r_p(p,\xi)}} + \frac{r_d^2(p,\xi)}{\sigma^2_{r_d(p,\xi)}} \right\|_\delta$

Figure 1. Overviewover the complete LSD-SLAM algorithm

2.2 Occupied grid mappreparation

Semi-dense three-dimensional point cloud data can be obtained from the LSD-SLAM algorithm, and the main obstacles in the indoor environment can be identified, but the three-dimensional data still needs to be converted into a two-dimensional grid map to be able to use the A* algorithm for path planning . Suppose a total of n three-dimensional data points, recorded as a set:

$$A = \{\, a_1(x_1, y_1, z_1)\,, \dots, a_n(x_n, y_n, z_n)\,\}$$

Specific steps are as follows:

STEP 1:

Filter out point cloud data that formed by objects higher than the robot itself, this objects are hanging on the robot some, such as lights, ceilings, etc., the robot can't reach higher than itself. To ensure that the camera is installed horizontally, according to the pinhole imaging model, the data with the y-axis less than 0 is removed and obtain the point set:

$$B = \{\, b\,(x, y, z) |\; b\; \in A \text{ and } y \geq 0\,\}$$

Suppose the number of points left nowis$m$, then B can be described as:

$$B = \{\, b_1(x_1, y_1, z_1)\,, \dots, b_m(x_m, y_m, z_m)\,\}$$

STEP 2:

$\forall\, b \,\epsilon\, B\,,\;\; y = 0$, and all points are projected onto the same plane. Then a new set of points is formed:

$$P = \{\, p_1(x_1, 0, z_1)\,, \dots, p_m(x_m, 0, z_m)\,\}$$

Suppose the number of horizontal grids is H and the number of vertical grids is$V$, then the grid points can be represented as point sets:

$$Q = \{\, q\,(h, v) |\; 0 \leq h \leq H, 0 \leq v \leq V, h, v\, \epsilon\, \mathbb{N}\,\}$$

There is a mapping $f: P \rightarrow Q$between$P$and$Q$,seeequations (1).

STEP 3:

After making P as Q, the grid q_i will containtotal l points internally, as shown in Fig.2(a),blue points are clouds points. Select a threshold T_1, when $l \geq T_1$, $q_i = 1$, which means that the grid can'treach, when $l < T_1$, $q_i = 0$, means the grid can reach, so that you can make into OGM map, as shown in Fig.2(b), the black grid can't reach, white grid isreachable area.

$$f := \begin{cases} X_{max} &= MAX\{x_1, x_2, \dots, x_m\} \\ X_{min} &= MIN\{x_1, x_2, \dots, x_m\} \\ u &= \frac{X_{max} - X_{min}}{H} \\ Z_{max} &= MAX\{z_1, z_2, \dots, z_m\} \\ Z_{max} &= MAX\{z_1, z_2, \dots, z_m\} (1) \\ v &= \frac{Z_{max} - Z_{min}}{V} \\ p_i(x_i, z_i) \in P, q_i(h_i, v_i) \in Q & \\ h_i &= \left\lfloor \frac{x_i - X_{min}}{u} \right\rfloor \\ v_i &= \left\lfloor \frac{z_i - Z_{min}}{v} \right\rfloor \end{cases}$$

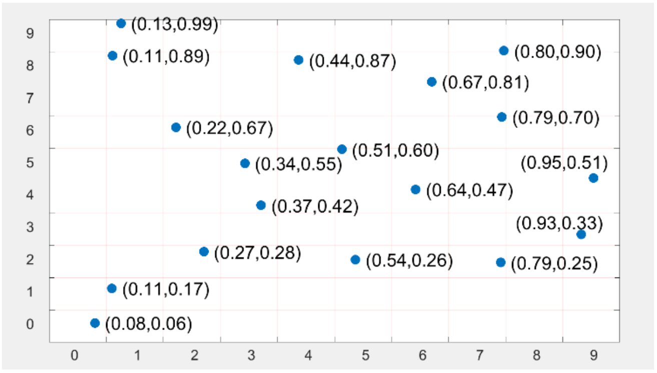

Figure 2(a). Every grid contain some points**.**

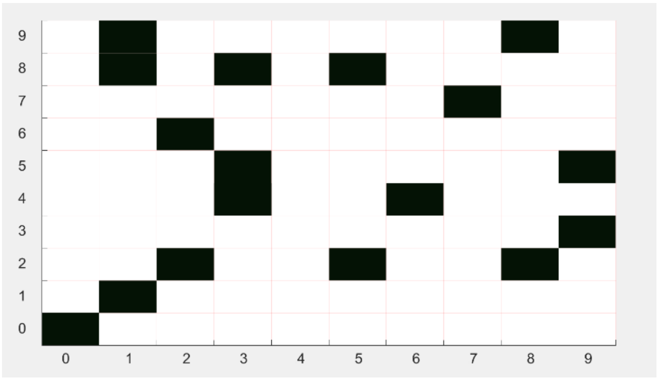

Figure 2(b). Occupied grid map.

2.3 Occupied grid mapexpansion

After obtaining OGM map, the robot itself has a certain size, but in the path planning, it is to use a grid to represent the robot, so the robot may collidesome obstacles. It is necessary to expand the obstacles before proceeding with the path planning, update to the OGM map, and then do the pathplanning. For acertain non-accessible grid, expansion area to the center of the circle, T_2 radius of the circular area.

3 Path planning

3.1 A* algorithm

The A* algorithm is a commonly used heuristic path planning algorithm that can be applied to find the shortest path from the start to the end of a grid map. The algorithm relies on the evaluation function $f(n)$ to measure whether the path is optimal.

$$f(n) = g(n) + h(n) \qquad (2)$$

The evaluation function $f(n)$is an evaluation function of node n, $g(n)$represents the movement cost evaluation of node n from the starting point, which is the movement distance of all the passed parent nodes by adding n to the starting point; $h(n)$is cost to the end point, which is a heuristic value where the Manhattan distance is used to guide the algorithm to find the end point. As shown in Fig.3, the redletter S indicates the start node, the green letter E indicates the end node and the blue node indicates the found path. Algorithm pseudo-code shown in Fig.4.

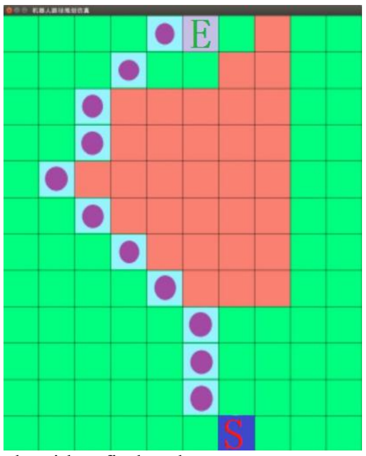

Figure 3. A* algorithm find path.

```
Input  : Location of Start and End node
Output : Father node of End

1. open          =  list containing Start node;
2. closed        =  empty set;
3. movecost(x;y) =  distance from node x to node y.
4. while End node not in open : do
5.     i = node with lowest f (i) in open ;
6.     remove i from open ;
7.     add i to closed ;
8.     count = 0;
9.     for j = neighbor node of i and not in closed and reachable do :
10.        count++ ;
11.        cost = g(i) + movecost(i;j);
12.        if j in open and cost < g(j) then :
13.            remove j from open ;
14.        if j not in open and not in closed then :
15.            add j into open ;
16.            f (j) = g(j) + h(j);
17.            set father node of j is i;
18.    if count == 0 then :
19.        can’t ¯nd path;
20.        break out;
```

Figure 4. A* algorithm pseudo-code.

3.2 A* algorithm performanceimprovement

We perform three main operations on the open set: the main loop finds the best node in the open set and adds it to the closed set, checks whether it is in the open set while visiting the neighbour node, and inserts the new node into the open set. Generally, the list is used to store the open set, the time complexity of deleting the insertion operation is $O(n)$, and here the binary heap is used to store the open set, which can reduce the time complexity to $O(\log n)$.

4 Simulation and experimental

4.1 Mapping using LSD SLAM

The physical prototype used in this paper is omnidirectional wheel mobile platform. The physical prototype has three 60mm omnidirectional wheels driven by MD36 type 24V DC motor, self-developed CAN bus DC servo motor driver and open UART and CAN motion control interface to the upper computer equipment. Using HTPC industrial computerforimage progressing and obstacle avoidance calculation. Using MT9V034 type camera, global shutter, angle of view 140°, focal length 1.9mm, resolution 640×480 , frame rate 60Fps, complete visual collection; Using RPLIDAR A1 type Lidar, 360° scan, 6mmeasurement radius, the data acquisition frequency of 5.5HZ ,avoid obstacles. The overall electrical connection is shown in Fig.5(a), and the robot entity is shown in Figure Fig.5(b).

Controlling the physical prototype to complete the loop-back motion in the indoor environment, as shown in Fig.6(a), the visual odometer calculates the pose change of $\mathfrak{se}(3)$, and optimizes the map and Open FAB-Map to complete the loopback detection using g^2o . Finally, 639007 3D point cloud data are obtained, and an ASCII version of the polygon file format document (PLY) is saved. The obtained 3D point cloud map and the pose diagram of the physical prototype are shown in Fig.6(b).

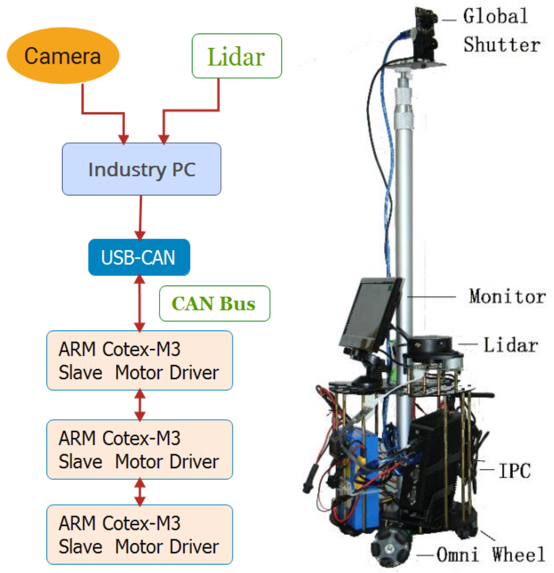
Camera
Lidar
Industry PC
USB-CAN
CAN Bus
ARM Cotex-M3 Slave Motor Driver
ARM Cotex-M3 Slave Motor Driver
ARM Cotex-M3 Slave Motor Driver
Global Shutter
Monitor
Lidar
IPC
Omni Wheel

Figure 5 (a). Electrical Connection**(b).** Robot

4.2 Occupied grid mappreparation

After obtaining the PLY file, 364413 points cloud data are obtained according to STEP1 of Section 2.2, and then all the points are projected onto the ground to obtain a scattergram. Fig.7(a) is a path planning software interface running by the industrial computer.The software is based on OpenGL set up to achieve real-time display of 2-D terrain data, the number of horizontal grid is 168, the number of vertical grid is 120, the red point is the point of cloud projection, accounting for one pixel. Progressing the map according to STEP 2 and get the grid map as shown in Fig.7(b). The red area is not accessible and the green area is the accessible area. There are 8007 unacceptable grids and 121153 grids available.

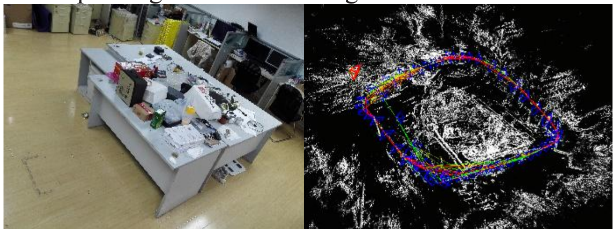
Figure 6 (a). Indoor room**(b).** Point clouds map

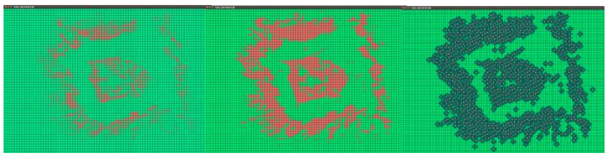
Figure 7 (a). Scatter plot**(b).** OGM.**(c).** Expansion OGM

Then the expansion of the grid points is performed, using a Bresenham drawing circle algorithm from computer graphics--using a series of discrete points to approximate the circle and obtaining a grid expansion with a radius of 2 as shown in Fig.7(c). Radius can be dynamically adjusted

4.3 Path Planning Algorithm Simulation

To evaluate the performance and performance of the A* algorithm, the algorithm was tested in the simulation software described above. The author first tests the performance of two data structure A* algorithms at different start node andend node on different size maps of $168 \times 120, 336 \times 240, 672 \times 480$,as shown in Fig.8(a) and (b). The yellow grid indicates the path to the plan. The letter S indicates the start node and the letter E indicates the end node. Fig.9 shows the comparison of the three methods at different fifty pairs of start node andend node. The vertical axis shows the fifty pairsaverage time.Unit issecond. It can be seen that the time cost of A* binary heap is about 1/5 of the A* list, while the Dijkstra algorithm has the longest time, ten times as much as the A* list.

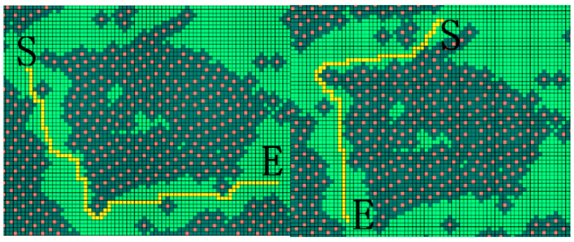

Figure 8 (a). West to east**(b).** North to south

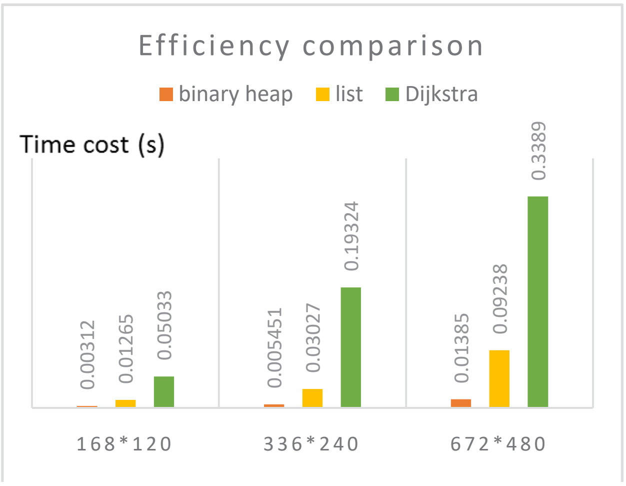

Figure 9. Vertical axis is average of fifty different pairs of start node and end node summary time cost. Horizontal axis is three different maps. Orange is time cost of A* using binary heap data structure, yellow is time cost of A* using list data structure, green is time cost of Dijkstra algorithm.

5 Conclusions

Inthis paper, the grid map preparation method and path planning strategy based on monocular vision simultaneous positioning and map construction are studied.

The environment is reconstructed using the large-scale direct method (LSD) model, which enables the environment to be accurately reconstructedin various scenarios of robots. Based on the environment three-dimensional data, occupancy grid map (OGM) was prepared for the description of the terrain. According to the terrain information, A * algorithm was used for path planning, and the performance of the algorithm was improved. Simulation shows that the proposed method is more than 5 times faster than the traditional method.

Next research will focus on controlling robot driving along the path and fixing robot pose according to VSLAM $\mathfrak{se}(3)$ pose.

References

1. E.W.Dijkstra, *A Note on Two Problems inCinnexion with Graphs. Numerische Mathematik*,1,269-271,(1959).
2. Amit Patel *Introduction to A** ,1-28,(1997-10-16)
3. Ferguson D,StentzA,Using interpolation to improve path planning: the field D algorithm. Journal of Field Robotics, 2006, 23(2) : 79-101.
4. XIN Yu,LIANG Huawei,DU Mingbo,MEI Tao, WANG Zhiling,JIANG Ruhai,An Improved A* Algorithm for Searching Infinite Neighbourhoods. Robot,(20-14),36(05):627-633.
5. LI Chong,ZHANG An,BI Wenhao.Single-Boundary Rectangle Expansion A* Algorithm.Robot,2017,39 (01):46-56
6. XiangGao,TaoZhang,YiLiu,QinruiYan.Fourteen Lectures on Visual SLAM: From Theory to Practice

[M]. Beijing: Publishing House of Electronics Industry, 2017: 130-137
7. Georg Klein, David Murray. Parallel Tracking and Mapping for Small AR Workspaces. In Proc. International Symposium on Mixed and Augmented Reality (ISMAR 2007, Nara)
8. Raúl Mur-Artal and Juan D. Tardós. ORB-SLAM2: an Open-Source SLAM System for Monocular, Stereo and RGB-D Cameras. *IEEE Transactions on Robotics,* vol. 33, no. 5, pp. 1255-1262, (2017)
9. J. Engel, J. Sturm, D. Cremers, Semi-Dense Visual Odometry for a Monocular Camera, J. Engel, J. Sturm, D. Cremers, ICCV (2013)
10. J. Engel, T. Schöps, D. Cr*emers*, LSD-SLAM: Large-Scale Direct Monocular SLAM, ECCV(2014)
11. J. Engel, V. Koltun, D. Cremers.Direct Sparse Odometry , In arXiv:1607.02565, (2016)
12. R. Wang, M. Schwörer, D. Cremers Stereo DSO: Large-Scale Direct Sparse Visual Odometry with Stereo Cameras , In International Conference on Computer Vision (ICCV), (2017)
13. V. Usenko, J. Engel, J. Stueckler, D. Cremers, Reconstructing Street-Scenes in Real-Time From a Driving Car , In Proc. of the Int. Conference on 3D Vision (3DV), (2015)
14. J. Engel, J. Stueckler, D. Cremers,Large-Scale Direct SLAM with Stereo Cameras , In-International Conference on Intelligent Robots and Systems (IROS), (2015)
15. Arren Glover,William Maddern,OpenFABMAP: An Open Source Toolbox for Appearance-based Loop Closure Detection,ICRA,14-18 May (2012)