IECON2015-Yokohama
November 9-12, 2015

Mobile Robot Navigation in Unknown Corridors using Line and Dense Features of Point Clouds

Kun Qian, Zhijie Chen, Xudong Ma, Bo Zhou
Key Laboratory of Measurement and Control of CSE, Ministry of Education,
School of Automation, Southeast University
No.2, Sipailou, Nanjing 210096, China
kqian@seu.edu.cn

Abstract—**This paper addresses the problem of mobile robot navigation in unknown corridors using RGB-Depth cameras. Instead of building a full and global 3D map of the environment, the approach exploits line and dense features extracted from RGB-D sensors. Wall-floor boundary lines are extracted from pre-processed point clouds, which ensure reliable line segmentation results compared with monocular based methods. A strategy is then proposed to compute the reference tracking points along the corridor for a wall-following behaviour. Meanwhile, dense 3D point clouds with ground-plane removed are projected which provide occupancy information, so that existing obstacle avoidance algorithms can be reused. A goal-directed navigation function is also developed by constructing a Nearness Diagram based obstacle avoidance behaviour guided by a wall-following behaviour. Experiment results validate the practicability and effectiveness of the approach.**

Keywords—**RGB-D Sensor; Floor plane extraction; RANSAC; Obstacle Avoidance; Mobile Robot**

I. INTRODUCTION

To support mobile robot navigation in typical indoor environments such as corridors, vision-based methods usually make use of walls, floors, door frames and other parts of architecture that provides rich geometrical features. By knowing where the floors and walls are, robots can avoid obstacles by navigating within the free space.

A significant amount of research has focused upon the vision-based avoidance problem. In these techniques, detecting floors and walls using a single camera has been a popular method[1][2]. Many methods rely upon homography to reconstruct the floor plane information from multiple cameras. In paper [1], the authors have proposed a technique that combines three visual cues from single images for evaluating the likelihood of horizontal intensity edge line segments belonging to the wall-floor boundary. However, wall-floor boundaries in corridor images are difficult to determine due to strong reflections and shadows.

Recently, low-cost RGB-D sensor[3][4][5][6][7] (e.g., Kinect) has become an attractive alternative to expensive laser scanners in robotic application areas such as indoor mapping and human-robot interaction. RGB-D sensors allow affordable and easy computation of depth maps. Some researches about the performance of a RGB-D camera in environment modeling have achieved successful results[8]. To allow ground plane fitting when the point cloud includes other planes, RANSAC plane fitting algorithm[9][10] can be applied to remove large amount of outliers.

Although RGB-D sensors have been successfully applied to 3D environmental modeling, the 3D dense map cannot be effectively applied to robot navigation directly.

There are two major problems concerning RGB-D sensor based robot navigation in unknown indoor environment such as corridors. Firstly, 3D point clouds contain large amount of redundant data for real-time processing, which is crucial to robot navigation. The other problem is that classic obstacle avoidance algorithms require the 3D point clouds to be projected on the 2D ground plane[12]. Thus how to extract useful environmental features from RGB-D perceptions is also an important factor that determines the effectiveness and reliability of robot navigation. In typical indoor environments, wall-floor boundaries and other line features are useful for a robot to decide the goal in the free space to traverse. Meanwhile, obstacle occupancy probabilities also provide rich information for obstacle avoidance. When using RGB-D perceptions, how to combine these two sources of information remains a problem to be solved.

Map based path planning is a usual solution when a robot navigation in unknown environments. In [11], the authors propose a real-time RGB-D based localization and navigation method, which makes it possible to reuse existing 2D maps. But in this paper we are concerned with the problem of robot navigation in unknown environments without a given map or the need of building a global map. Usually, building accurate maps for corridor environments is difficult, since the walls cannot provide enough features to be distinguished for most 3D SLAM algorithms.

To cope with the above mentioned problems, in this paper we have proposed a new method of mobile robot corridor navigation and obstacle avoidance using line and dense features extracted from point clouds. Our contribution is a methodology that uses both line and dense information extracted from noisy point clouds without the use of color information to ensure more reliable navigation robots in situations when the color features of walls, floors and obstacles are not significant enough.

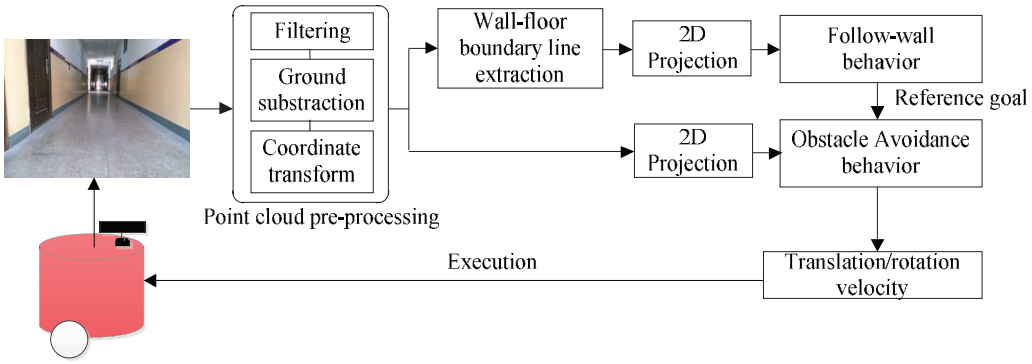


Fig. 1. The method outline

II. The System Outline

The method outline is shown in Fig.1. The feature extraction procedure includes the use of geometrical features of corridors and the probabilistic grid occupancy of the cells on the ground plane of the corridor.

In particular, the wall-floor boundary lines are extracted by using Hough transform[13] on the 3D point cloud readings after a pre-processing step. These 3D point clouds are projected onto a two-dimensional grid through plane projection, which provides wall-floor boundray and obstacle measurements in robot's local coordinate frame. The obstacle avoidance behavior is developed based on Nearness Diagram algorithm[14] to generate reactive obstacle avoidance actions. This obstacle avoidance behavior is guided by the wall-following module towards a reference tracking point.

III. Line Features And Dense Map Projection

A. Point cloud pre-processing

The algorithm begins with point clouds pre-processing which includes three major steps: filtering, segmentation and coordinate transform.

Firstly, a voxel grid filter is applied to reduce the amount of data obtained as much as possible while maintaining the desired features of the input raw readings. The filtering step greatly reduces the computational load of further processing.

Secondly, the point cloud is segmented into several different planes using RANSAC algorithm[15]. A plane is described as the following equation:

$$ax + by + cz + d = 0 \tag{1}$$

in which (a,b,c) is the coefficients of the plane and d is a threshold. The RANSAC algorithm iteratively adjusts the parameters of the plane which contains the highest quantity of points. Let axis x and z be in the horizontal plane and y be vertical to the ground plane. Assume that the sensor is fixed on the robot with the optical axis parallel to the floor, a plane that satisfies $a,c \approx 0$ and $b \approx \pm 1$ is considered to be horizontal, since the height of the plane is constant, i.e., $y \approx -d/b$. Therefore, the plane is labeled as floor if $y < 0$.

Thirdly, the point clouds from camera coordinates are transformed to world coordinates, so that the ground plane became the X-0-Y plane (0*x+0*y+z=0). Denote (K_x, K_y, K_z) as a point in the camera coordinate and (W_x, W_y, W_z) is the corresponding point in the world coordinate, Equation (2) is applied, in which T and θ are the relative translation and angular rotation between the Kinect camera coordinate and the world coordinate. As the result of the coordinate transform, points that are located on the same horizontal plane have the same z value. Fig. 2 shows the relation between different coordinates.

$$\begin{bmatrix} W_X \\ W_Y \\ W_Z \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & T_X \\ 0 & \cos\theta & -\sin\theta & T_Y \\ 0 & \sin\theta & \cos\theta & T_Z \end{bmatrix} \cdot \begin{bmatrix} -1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} K_X \\ K_Y \\ K_Z \\ 1 \end{bmatrix} \tag{2}$$

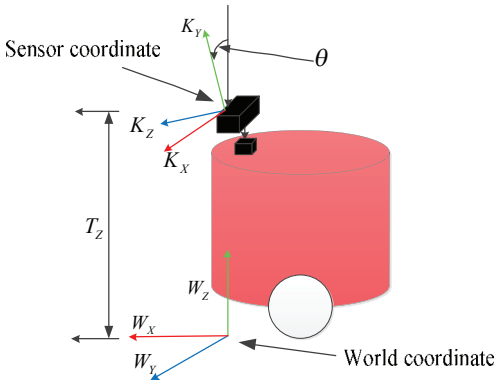
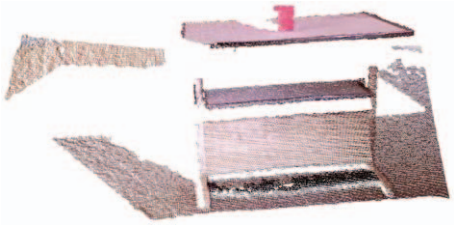
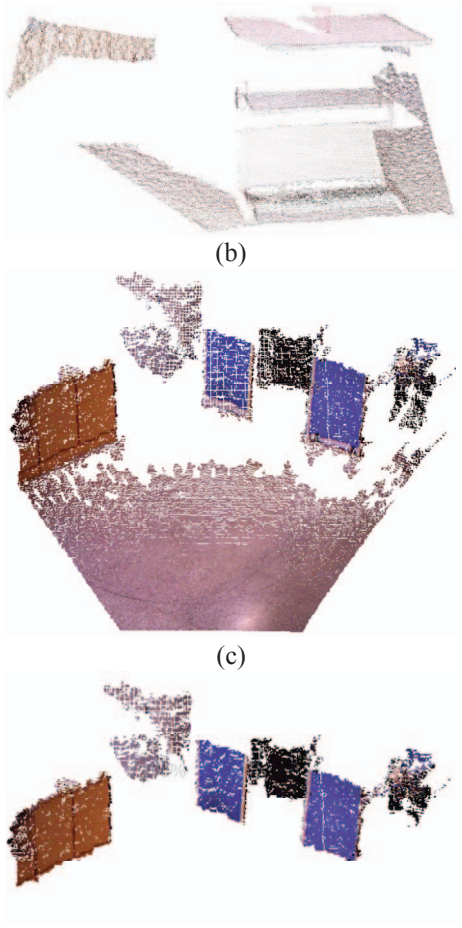


Fig. 2. Coordinate transform

The result of point cloud pre-processing is a 3D scan from a fixed horizontally aligned virtual depth sensor at the base of the robot. In addition, for the further corridor line segmentation, the pre-processing avoids some future problems due to the excess of line information contained in the point cloud, and thus improves its speed. Fig.3 is the result of point cloud pre-processing, in which Fig. 3(a) is an example of raw point cloud, Fig. 3(b) is the result of filtering, Fig. 3(c) is another example of raw point cloud, and Fig. 3(d) is the result of ground-plane removal.



(a)



(b) (c) (d)

Fig. 3. Point cloud pre-processing example

B. Wall-floor boundary line extraction

After the pre-processing step, we use Hough transform on the resulting point cloud data to extract the lines of the wall-floor boundary lines (i.e., skirting lines).

A line can be expressed by two variables (ρ,θ) in the polar coordinate system, i.e., the line function is $\rho = x\cos\theta + y\sin\theta$. Hough transform quantizes the parameter space of (ρ,θ) by n bins,

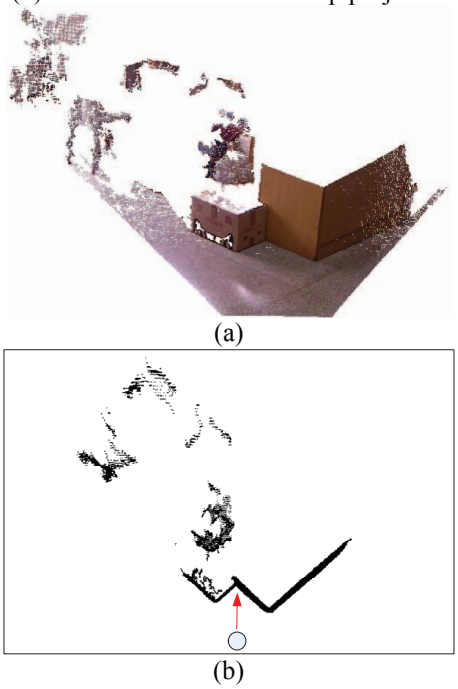
$$\begin{cases} \rho_n = (\mathrm{n}-1/2)\Delta\rho & n = 1,2,...N_\rho \\ \theta_n = (\mathrm{n}-1/2)\Delta\theta & n = 1,2,...N_\theta \end{cases} . \quad (3)$$

in which $\Delta\rho = L/N_\rho$, $L = \max(\sqrt{x^2+y^2})$, $\Delta\theta = \pi/N_\theta$, N_ρ and N_θ is the segment number of ρ and θ , respectively. Hough transformation for detecting straight lines is through a point-to-curve transformation. The transform is implemented by quantizing the Hough parameter space into finite intervals. As the algorithm runs, each line segment is transformed into a discretized (ρ_i,θ_i) curve and the accumulator cells which lie along this curve are incremented. Resulting peaks in the accumulator array represent strong evidence that a corresponding straight line exists.

Since the wall-floor boundary lines are below the Kinect sensor, in order to reduce the computation load, only the vertically lower half of the point cloud is analyzed for extracting the wall-floor boundary lines.

C. Map projection

Two sources of point clouds are projected on the ground plane. One source is the extracted point cloud that corresponds to the skirting line of both sides of the walls. The projected map only contains wall-floor boundary lines (no obstacles will be included), which provide key information for deciding a wall-following behavior. The other source is the remaining point cloud after the pre-processing step. In order to use the traditional class of obstacle avoidance algorithms that compute collision free paths in 2D occupancy grid maps, the 3D point cloud is projected onto the ground plane, which yields similar occupancy grid map as obtained by traditional range sensors. The parameters of the RANSAC algorithm used for floor plane extraction are also utilized for performing the projection. Fig. 4(a) shows an example of 3D point cloud and Fig. 4(b) shows the result of 3D map projection.



(a) (b)

Fig. 4. Local projected 2D map

IV. Corridor Navigation Strategy

A. Follow-wall behavior

The robot navigates through a corridor while keeping a certain distance to one side (take right side as an example) of the wall. The width of the corridor is TD_o . As an example, the testing corridor is about 15m in length and 2.5m in width. The robot is assumed to be initially placed at a location where the distance to right side of the wall is td . Then the parallel line to the right wall-floor boundary line is assumed to be the reference tracking line.

When the extracted skirting line in 3D point clouds are projected onto the 2D ground plane, the ground plane coordinate system (x-y) can be illustrated as in Fig.5. If we

consider a given depth y' in front of the robot, in the (x-y) plane, (x'_l, y') and (x'_r, y') are the corresponding points at the left and right wall-floor boundaries, respectively. The reference tracking line intersects with the virtual horizontal line at (x'_{ref}, y'). The reference tracking point (x_{ref}, y_{ref}) is taken from the reference tracking line with a certain forward distance ahead.

In the (x-y) plane, the left and right wall-floor boundary line is modeled as the following equation, respectively:

$$\begin{cases} left: & y_l = k_l x + b_l \\ right: & y_r = k_r x + b_r \end{cases}. \quad (4)$$

Then the equation of the reference tracking line is:

$$x_{ref} = \frac{td \cdot x_l + (TD_o - td) x_r}{TD_o} = \frac{td\left[(y_l - b_l)/k_l\right] + (TD_o - td)\left[(y_r - b_r)/k_r\right]}{TD_o}. \quad (5)$$

The points on left and right boundaries with the same depth have the same y value, which is denoted as y_{ref}. So if we let:

$$k_{ref} = \frac{td/k_l + (TD_o - td)/k_r}{TD_o}, \quad (6)$$

$$b_{ref} = -\frac{td \cdot b_l/k_l + (TD_o - td) b_r/k_r}{TD_o}, \quad (7)$$

we will then have :

$$x_{ref} = k_{ref} y_{ref} + b_{ref}. \quad (8)$$

A reference tracking point is computed on the reference tracking line with certain look-ahead distance. We model the following three types of situation to compute the yaw angle and the horizontal offset distance.

The first situation is when the robot is not exactly on the reference tracking line and it's not parallel with the reference tracking line, i.e., $0 < k_l = k_r < \infty$. A Kinect sensor works most effectively when the depth measurement is 0.5m to 3.5m. Hence we choose a desired look-ahead y value of the reference tracking point, $y_{ref} = 1.6$m. Thus the desired x value of the reference tracking point, x_{ref}, can be easily computed from Equation(8). The resulting point (x_{ref}, y_{ref}) is then taken as the local goal point (x_{goal}, y_{goal}).

The second situation is when the robot is already parallel to the reference tracking line, i.e., $k_l = k_r = \infty$. In this case, we set $y_{goal} = 1.6$ and $x_{goal} = x_r - td$. In fact, this is the situation that the robot navigates toward the vanishing point of the corridor.

The third situation is when the robot's heading is vertical to the walls, $k_l = k_r = 0$. In this case, we have $x_{goal} = 1.6$ and $y_{goal} = y_l + TD_o - td$, if the robot's heading is vertical to the left wall. Similarly, we have $x_{goal} = -1.6$ and $y_{goal} = y_r - td$ if the robot's heading is vertical to the right wall.

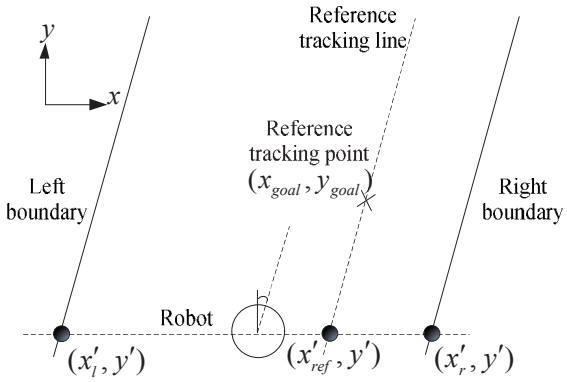


Fig. 5. Computation of reference tracking point

In all of the above three situations, the resulting (x_{goal}, y_{goal}) is marked in the current local perception window of the robot, which guides the obstacle avoidance behavior.

B. Obstacle avoidance behavior

In order to provide a sensory-based obstacle avoidance behavior whilst the robot performs the follow-wall behavior, we have constructed a Nearness Diagram (ND) based obstacle avoidance behaviour guided by a wall-following behaviour.

In particular, the follow-wall behavior control module calculates an optimal reference point along the corridor axis based on mapped obstacles. The ouput reference tracking point in robot's local perception window is considered to be the reference goal point for the obstacle avoidance control module. In the obstacle avoidance module, the Nearness Diagram (ND) algorithm is employed based on the local projected 2D map that is updated by the sensor.

The reactive navigation module creates two types of nearness diagrams, namely "ND from the central Point" (PND) and "ND from the Robot bounds" (RND)[13]. The Nearness Diagram based local reactive obstacle avoidance controller computes actual translational velocity v and rotational velocity w based on the relation of the reference points and the shape of obstacles. The ND algorithm defines a set of situations that are represented in a decision tree according to a high safety criteria and a low safety criteria. Denote $v_{\max}$ as the maximum robot traversing velocity, $w_{\max}$ as the maximum robot rotational velocity, d_b as the nearest obstacle to the robot's boundary, and d_s as a safe distance, hence the output robot translation velocity v and the robot rotation velocity w are computed as:

$$\begin{cases} High\ Safety: & v = v_{\max} \cdot Abs(1 - 2\theta_v/\pi) \\ Low\ Safety: & v = v_{\max} d_b/d_s \cdot Abs(1 - 2\theta_v/\pi) \end{cases} \quad (9)$$

$$w = 2w_{\max} \theta_v/\pi \quad (10)$$

in which $\theta_v \in [-\pi/2, \pi/2]$ is the robot's desired direction computed by the ND algorithm.

V. EXPERIMENTS

The proposed approach entails the use of a mobile robot in a real office environment containing several rooms and a corridor that connects the rooms. *ActivMedia* Pioneer 2-AT/3-DX robots were used. The robot is equipped with a Kinect sensor mounted onboard with 39cm above ground level, as shown in Fig. 6. The developed RGB-D sensing and map building software is running on a laptop computer with Intel Core Duo CPU and Ubuntu 11.04. The testing corridor is about 15m in length and 2.5m in width.

Fig. 6. The robot platform

The robot involved in the experiment was initially placed about 0.6m to the right wall of the corridor and its traversing speed was set as 0.75m/s. The robot's onboard Kinect sensor was mounted onboard with 39cm above ground level and the pitch of the sensor is 12° to the ground plane.

In a testing scenario, the robot was set to follow the right wall of the corridor and keep 0.3m to the right wall. In the meantime, the robot's autonomous capability avoided the obstacles (e.g., boxes placed on the corridor ground plane).

Fig. 7 shows a snapshot of the testing results. In Fig. 7, ①denotes the place where the robot was frontally faced with the obstacle, ②denotes the situation that the robot turned left to avoid the possible collision with the object, and ③denotes the situation that the robot detoured and continued to follow the wall along the reference tracking line. Fig. 8(a) gives the extracted wall-floor boundary of the corresponding three situations. In Fig. 8(a), the upper line of sub-figures are the 3D point clouds with the wall-floor boundary lines marked in red color, the lower line of sub-figures illustrate the point cloud of the skirting line. Fig. 8(b) shows the projected 2D map of each corresponding situation, in the robot's local coordinate frame. According to the result, the robot successfully navigated through the corridor and avoided the obstacles.

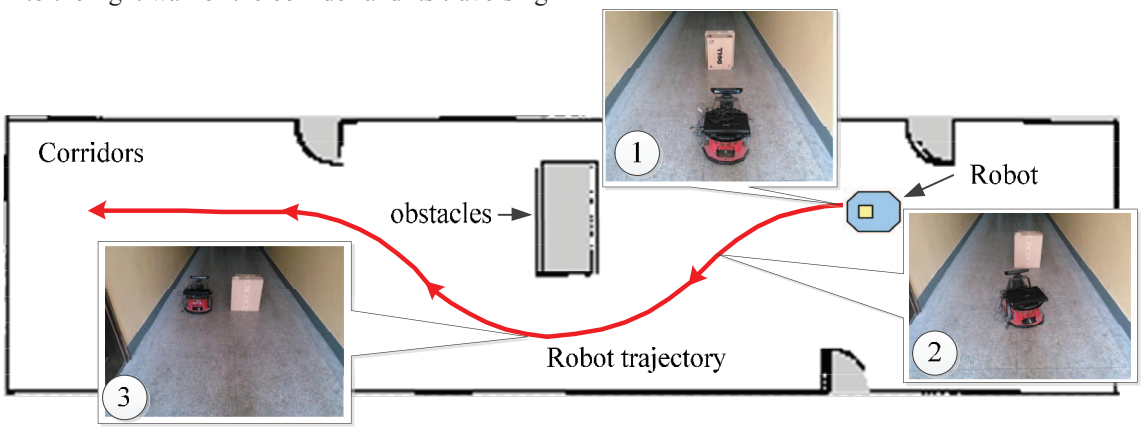
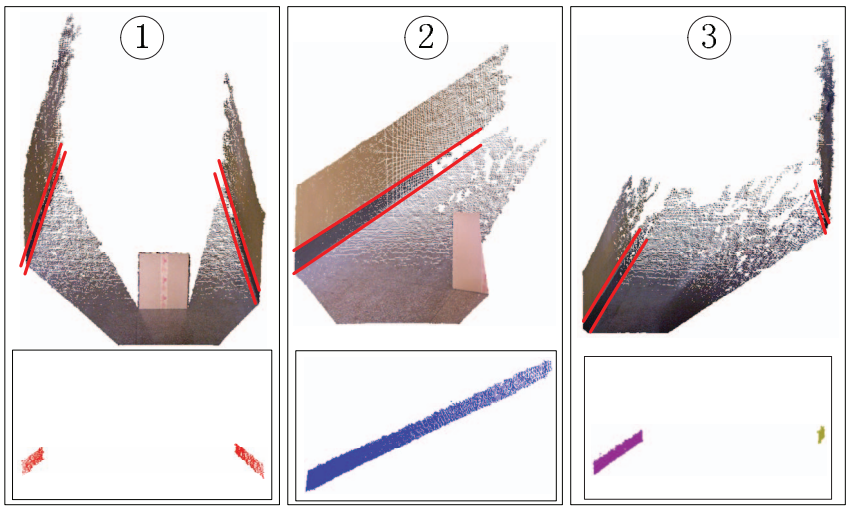
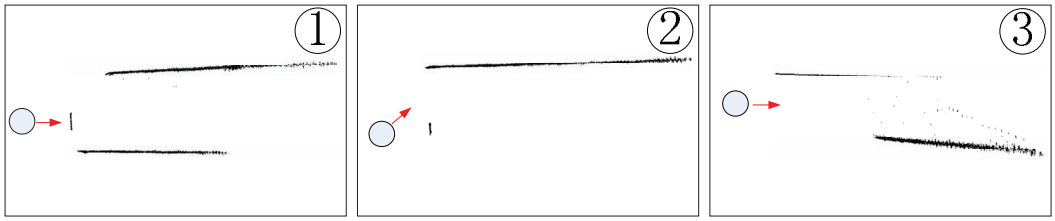


Fig. 7. The testing scenario



(a)The wall-floor boundary lines extracted



(b)The projected 2D map

Fig. 8. The experimental results

To test the reliability of the proposed method, we also conducted the corridor navigation experiment in two other testing corridors under different lightening conditions. We judge the success of a trial by testing if the robot can successfully traverse through the corridor from the entrance at one side to the exit at the other side. In our trial study, we found that in both day light experiments and night experiments, the proposed method ensured high success rate of corridor navigation. But in a special case that the robot was directly exposed to sunlight through windows at a noon time and the light was bright, the methodology cannot ensure sufficient successful rate, because sunlight can greatly reduce the accuracy of Kinect's reading, which is a known drawback of Kinect sensor[16]. Table 1 gives some trail study results. The failed cases are majorly caused by the lightening reasons as mentioned above.

TABLE 1. Trial study results

Experimental condition	Trial number	Success times	Fail times
Day light(no direct sunlight)	30	27	3
Day light(direct sunlight)	30	3	27
At night	30	28	2

VI. Conclusion

In this paper, a practical method of robot navigation in corridors is proposed, which combines the use of geometrical features of the environments and probabilistic occupancy information obtained from a low-cost Kinect sensor equipped on the mobile robot. The proposed method doesn't require a previously established map of the corridor environment and is suitable to be applied in unknown environments. The proposed method has two-fold advantages. On one side, it uses depth data for extracting wall-floor boundary lines, which ensures better reliability compared with monocular methods, especially in situations that the images are influenced by strong reflections or the appearance features of walls, floors and obstacles are not significant. On the other side, the method exploits the use of projected point cloud, so existing obstacle avoidance algorithm can be reused, and the reduced computational load enables real-time processing for robot navigation with high reliability.

Acknowledgment

This work is supported by the National Natural Science Foundation of China (Grant No. 61105094, No. 61573101, No. 61573100), the Fundamental Research Funds for the Central Universities(No. 2242013K30004), and it's also funded by the Priority Academic Program Development of Jiangsu Higher Education Institutions.

References

[1] Y. Li, S. Birchfield, "Image-based segmentation of indoor corridor floors for a mobile robot", in: *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2010, 837-843.

[2] N. Ohnishi, A. Imiya, "Corridor Navigation and Obstacle Avoidance using Visual Potential for Mobile Robot", in: *Proceedings of the Fourth Canadian Conference on Computer and Robot Vision*, 2007,131-138.

[3] D. Maier, A. Hornung, M. Bennewitz, "Real-time navigation in 3d environments based on depth camera data", in: *Proceedings of the 12th IEEE-RAS International Conference on Humanoid Robots*, 2012: 692-697.

[4] D. Kırcahn, F. B. Tek, "Ground Plane Detection Using an RGB-D Sensor", in: *Proceedings of the 29th International Symposium on Computer and Information Sciences*, 2014, 69-78.

[5] C.J. Taylor, A. Cowley, "Parsing indoor scenes using RGB-D imagery", in *Robotics: Science and Systems*, July 2012.

[6] A. Oliver, S. Kang, B. C. Wünsche, et al. "Using the Kinect as a navigation sensor for mobile robotics", *Proceedings of the 27th Conference on Image and Vision Computing*, 2012, 509-514.

[7] Y. Zhou, G. Jiang, G. Xu and et al. "Kinect depth image based door detection for autonomous indoor navigation", *IEEE RO-MAN*, 2014, 147-152.

[8] P. Henry, M. Krainin, E. Herbst, X. Ren and D. Fox, "RGB-D mapping: Using Kinect-style depth cameras for dense 3D modeling of indoor environments", *International Journal of Robotics Research*, 2012, 31(5):647-663.

[9] D. Holz, S. Holzer, R. Bogdan Rusu, S. Behnke, "Real-Time Plane Segmentation using RGBD Cameras", in: *Proceedings of the 15th RoboCup International Symposium*, vol. 7416, pp. 307–317, 2011.

[10] M. Heracles, B. Bolder, and C. Goerick, "Fast detection of arbitrary planar surfaces from unreliable 3D data", in: *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2009, pp.5717-5724.

[11] J. Biswas and M. Veloso, "Depth Camera Based Indoor Mobile Robot Localization and Navigation", in: *Proceedings of IEEE International Conference on Robotics and Automation*, May, 2012, pp. 1697-1702.

[12] J. Cunha, E. Pedrosa, C. Cruz, A. J. R. Neves, N. Lau, "Using a depth camera for indoor robot localization and navigation", In: *Robotics Science and Systems (RSS) RGB-D Workshop*, 2011.

[13] Shapiro, Linda and Stockman, George. "Computer Vision", Prentice-Hall, Inc. 2001

[14] J. Minguez, L. Montano, "Nearness diagram (ND) navigation: collision avoidance in troublesome scenarios", *IEEE Transaction on Robotics and Automation*, 2004, 20(1): 45-59.

[15] M. A. Fischler, R. C. Bolles, "Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography". *Communication of the ACM*, 1981: 24(6): 381-395.

[16] C. Montella, T. Perkins, J. Spletzer, M. Sands, "To the Bookstore! Autonomous Wheelchair Navigation in an Urban Environment", *Springer Tracts in Advanced Robotics*, 2014: 92, pp. 249-263.