SLAM and Navigation in Indoor Environments

Shang-Yen Lin and Yung-Chang Chen

Department of Electrical Engineering, National Tsing Hua University,
30013 Hsinchu, Taiwan
ycchen@ee.nthu.edu.tw

Abstract. In this paper, we propose a system for wheeled robot SLAM and navigation in indoor environments. An omni-directional camera and a laser range finder are the sensors to extract the point features and the line features as the landmarks. In SLAM and self-localization while navigation, we use extended Kalman filter (EKF) to deal with the uncertainty of robot pose and landmark feature estimation. After the map is built, robot can navigate in the environment based on it. We apply two scale path-planning for navigation. The large-scale planning finds an appropriate path from starting point to destination. The local-scale path-planning fills up the drawbacks of the prior step, such as dealing with the static and dynamic obstacles and smoothing the path for easier robot following. Through the experiment results, we show that the proposed system can smoothly and correctly locate itself, build the environment map and navigate in indoor environments.

Keywords: SLAM, EKF, navigation, path-planning, obstacle avoidance, robot.

1 Introduction

In recent years, there is more and more attention on robotics research, especially the service robot industry all over the world. Many applications of the robotic technology have been developed, such as security robots, nursing robots, and so on. All of them need to navigate in an associated environment and locate themselves. There are four major problems needed to be overcome, which are self-localization, environment map establishment, path to destination detection and collision avoidance. However, these four problems are not independent, but mutually correlated. For example, once self-localization consists in error, it may cause the wrong map building. And the wrong map will cause self-localization in larger error. Furthermore, the wrong map or self-estimated location may induce path-planning failure.

Fortunately, a number of researches have worked on these challenges. Simultaneous localization and mapping (SLAM) method builds an environment map using only relative environment observation and using the map for robot localization at the same time. An unbiased map needs correct robot location and vice versa. Smith et al. [1] [2] used probabilistic model, a state vector describing robot location and landmark position with a covariance matrix describing their mutual uncertainty, and extended Kalman filter (EKF) to represent and estimate the spatial uncertainty. Durrant-Whyte et al. [3] [4] proposed a framework of SLAM. Doucet et al. [5] solved the SLAM problem by means of Rao-Blackwellized particle filter (RBPF).

Y.-S. Ho (Ed.): PSIVT 2011, Part I, LNCS 7087, pp. 48–60, 2011.

Montemerlo et al. proposed FastSLAM algorithm which tracks the landmarks by extended Kalman filter and estimate the robot's pose by Rao-Blackwellized particle filter. There are also many researches using different sensors for SLAM [6][7][8].

Also many researches discussed about path planning and obstacle avoidance. The path planning problems are typically approached using one of these two categories: search-based, sampling-based. The basic idea of search-based planning is using regular grid cells to represent the configuration space. The path planning problem is done by searching the grids and finding a point-to-point path from starting grid to the goal grid. Dijkstra [9] first proposed the Dijkstra's Algorithm, which solves the shortest path problem by breath-first search. A* algorithm [10] further uses an admissible heuristic to reduce the search region. The continuing improvements including D* algorithm [11] which makes re-plan more efficient, Anytime A* [12] which concerns the deliberation time and AD* [13] which combines D* and Anytime A*. The sampling based planning does not use the regular grid cells but samples the vertices in the configuration space with appropriate edge assignment between them, and finds path from the candidate vertices. The probabilistic roadmap (PRM) [14] generates vertex by random sampling. The rapidly-exploring random tree (RRT) [15] makes the sampling more efficient. There are also Anytime RRT [16] and Anytime Dynamic RRT [17] Algorithms proposed for improving the speed of planning and re-planning.

In this paper, we propose a system for wheeled robot navigation in indoor environments using only on-robot sensors. When the robot enters a new environment, we first build the environment map by SLAM (Simultaneous Localization and Mapping). In our approach, we choose the omni-directional camera and laser range finder as the on-robot sensors which have wide sensing field. The property of wide sensing field is very important for robot localization and obstacle capturing because of increasing the duration of landmarks and obstacles observation, and decreasing the effect of landmarks being covered by obstacles. After the environment map has been built, the robot can navigate by finding the appropriate path from the built map. We separate the robot navigation into two parts. The first part is the large-scale path-planning, which is similar as people select which path to go through. The other part is the local-scale path-planning for obstacle avoidance. This part rapidly generates a collision-free path and guarantees the robot real-time avoiding the obstacles when there are static or dynamic obstacles on the way to goal.

The remaining sections of the paper are organized as follows. Section 2 gives the overview of our system. Section 3 introduces feature extraction with omni-directional camera and laser range finder. The SLAM and localization method using the point and line features is presented in section 4. Section 5 presents the large scale and local scale path-planning for robot navigation. Experimental results are shown in Section 6. Finally, conclusion is presented in section 7.

2 System Overview

Figure 1 and Figure 2 show the flowcharts of our system. Figure 1 shows the SLAM flowchart, which is used to build the map when robot first enters a new environment. We utilize the omni-directional camera and SICK laser range finder to extract

landmarks. The landmarks used in our system are point features and line features, which will be briefly described in next section. We use the extracted features and odometer data for SLAM. After data association, we revise error with extended Kalman filter.

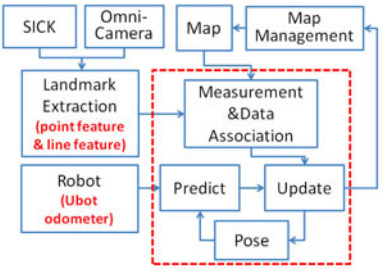

Fig. 1. Flowchart of SLAM

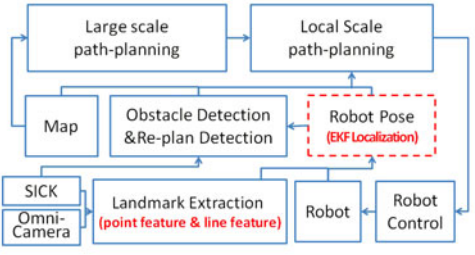

Fig. 2. Flowchart of robot navigation

Figure 2 shows the flowchart of robot navigation. After the environment map is built, robot can navigate based on it. When robot navigates in the indoor environment, we still use the same method as Figure 1 for robot self-localization. The dotted block" Robot Pose" in Figure 2 is same as the dotted block in Figure 1. For navigation, we first apply a large-scale path-planning. In this step, the algorithm finds an appropriate path from the location of robot to the destination. After the path is generated, robot can go along it and move toward destination. While the robot is moving, there may be some obstacles which block the original path as detected by the laser range finder. The local-scale path-planning can quickly generate a new collision free path for avoiding robot collision. Finally, the robot can safely achieve to the destination.

3 Landmark Extraction

We use an omni-directional camera and a laser range finder as our sensor system to extract the landmarks in the environment. It is easy to combine data of these two sensors because they are center-aligned, facing the same direction, and both sensing data can be represented in polar coordinates. In landmark extraction, we extract two types of landmarks: the point landmarks and the line landmarks. The point landmarks are used for both x-y location and robot orientation estimation, similar to many point-feature-based SLAM works. But the estimation with only point landmarks may have larger error if there are very few number of point landmarks in the observation region. Therefore, we add the line landmarks in our system. The line landmarks can improve the accuracy of robot orientation estimation.

3.1 Point Landmarks

A point landmark is the 2-D position of a 3-D vertical line, which is the intersection of the 3-D line with x-y plane as shown in Figure 3(a). It is very efficient to extract the vertical lines from images captured by onmi-directional camera, because of the property that all of the 3-D vertical lines extending pass through the center. Figure

3(b) shows the omni-directional camera data input. Because of the ratio of the image could influence the bearing information of the extracted landmark, we first resize the image to equal ratio, as shown in Figure 3(c). To make our processing concentrate on the useful region, we use a mask as given in Figure 3(d). Only the data in the white region will be processed. Then we apply the Canny edge detector to find out the edges. After we got the edge points, we record how many edge points exist in each angle degree. Once the number of existing edge points in an angle degree exceeds a threshold, a vertical line is affirmed. Combining with the laser range finder data, as the yellow points shown in Figure 3(f), we can get the position of the point landmarks. Finally, to reduce the observation error, we ignore those landmarks which are too far away. And for those landmarks existing in continuous angle degree, we only use the two sides of them as our point landmarks.

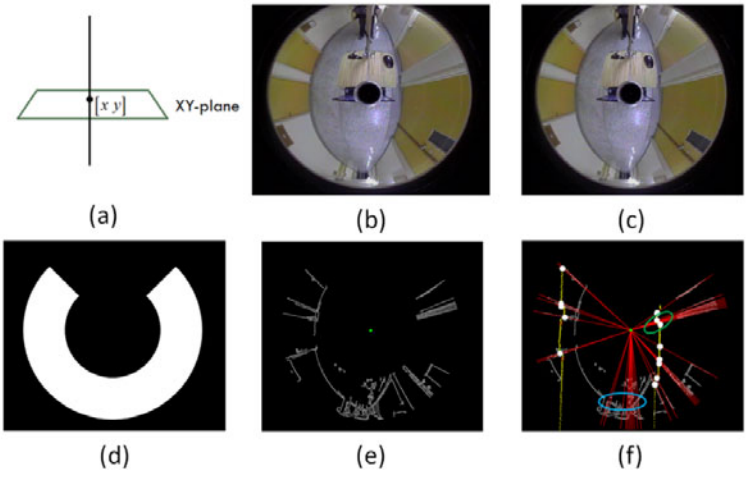

Fig. 3. Illustration of point landmarks extraction

3.2 Line Landmarks

The line landmarks are the horizontal straight lines in the environment, which are extracted from the laser range finder data. Figure 4(a) shows the laser range finder data input. For extracting the straight lines, we first find the break points for separating each straight line. For each range point s_i, if s_i satisfies one of the three conditions in (1), we consider s_i as a break point. These three conditions are considered for different situation. The first condition focuses on the distance between two continuous range points. If the distance is huge, as shown by red circles in Figure 4(b), two sides of s_i should not belong to same straight line. The second condition considers the angle between $\overrightarrow{s_{i-1}s_i}$ and $\overrightarrow{s_i s_{i+1}}$. A large angle should not occur on a straight line. The third condition is dealing with the situation shown by the green circles in Figure 4(b). We can see s_i in two green circles in Figure 4(b) have similar information of distance and angle, but those on the left form a straight line while

those on the right are break points. It is hard to use one threshold to distinguish them. Therefore, we use two smaller thresholds for double checking. After the break-point detection, those remaining range points could be seen as straight line points. For the sets consisting of consecutive points more than a threshold, apply the least square method and find the parameter of the line landmarks.

$$s_i \in \text{break point} \quad if \quad \begin{cases} dis(s_{i-1}, s_i) > \lambda_{d1} \;\|\; dis(s_i, s_{i+1}) > \lambda_{d1} \\ \left| dir(s_{i-1}, s_i) - dir(s_i, s_{i+1}) \right| > \lambda_{\theta 1} \\ \left(\begin{array}{c} (\lambda_{d1} > dis(s_{i-1}, s_i) > \lambda_{d2} \;\|\; \lambda_{d1} > dis(s_i, s_{i+1}) > \lambda_{d2}) \\ \&\& \\ \left| dir(s_{i-1}, s_i) - dir(s_i, s_{i+1}) \right| > \lambda_{\theta 2} \end{array} \right) \end{cases}$$

$$where \quad \lambda_{d1} > \lambda_{d2} \quad , \quad \lambda_{\theta 1} > \lambda_{\theta 2} \tag{1}$$

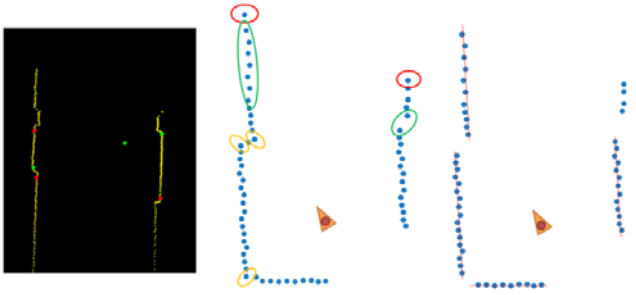

Fig. 4. Illustration of line landmarks extraction

4 Extended Kalman Filter SLAM

In SLAM, the uncertainty of the robot pose and landmark position is the dominant problem to be solved. In our system, we use EKF to handle uncertainties. The basic Kalman Filter is suited for a linear system. For a non-linear case we should linearize the original system appropriately. Generally, the EKF SLAM uses a state vector and a covariance matrix to describe status of the robot pose and landmark position. In our system the state vector is described as follows.

$$\mathbf{x}_k = \begin{bmatrix} \mathbf{x}_r & m_1 & \dots & m_n \end{bmatrix}^T \qquad \mathbf{x}_r = \begin{bmatrix} x_r & y_r & \theta_r \end{bmatrix}^T \qquad \mathbf{m}_i = \begin{cases} \begin{bmatrix} x_i & y_i \end{bmatrix}^T & \text{(point landmarks)} \\ \begin{bmatrix} \theta_i \end{bmatrix} & \text{(line landmarks)} \end{cases} \tag{2}$$

$\mathbf{x}_k$ is the state vector at time k. $\mathbf{x}_r$ is the robot pose vector, where $\begin{bmatrix} x_r & y_r \end{bmatrix}^T$ is the robot location in the world coordinate and θ_r is the orientation of the robot. $\mathbf{m}_i$ is the i^{th} landmarks as $\begin{bmatrix} x_i & y_i \end{bmatrix}^T$ for point landmarks and $\begin{bmatrix} \theta_i \end{bmatrix}$ for line landmarks.

- SLAM by EKF contains following main steps:

4.1 Prediction

We use the motion model and the previous state to predict the current state. The state prediction step is

$$\hat{\mathbf{x}}_{r\,k|k-1} = f(\hat{\mathbf{x}}_{r\,k-1|k-1}, \mathbf{u}_{k-1}) + q_k \tag{3}$$

$\mathbf{u}_{k-1}$ is the odometer data of motion, including velocity and angular velocity.

$$\mathbf{u}_k = \begin{bmatrix} V_k & \omega_k \end{bmatrix} \qquad \text{where } V = \frac{v_l + v_r}{2}, \omega = \frac{v_l - v_r}{l} \tag{4}$$

f is a non-linear function, describing how the robot moves in the world coordinate.

$$f(\hat{\mathbf{x}}_{r\,k-1|k-1}, \mathbf{u}_{k-1}) = \begin{bmatrix} x_{r\,k-1} + V_{k-1}\Delta T \cos(\theta_{r\,k-1}) \\ y_{r\,k-1} + V_{k-1}\Delta T \sin(\theta_{r\,k-1}) \\ \theta_{r\,k-1} + \Delta T\, \omega_{k-1} \end{bmatrix} \tag{5}$$

And the covariance matrix $P_{k-1|k-1}$ is propagated through the linearized state transition function f, yielding $P_{k|k-1}$ given by

$$P_{k|k-1} = J_f P_{k-1|k-1} J_f^T + Q_k \qquad \text{where } J_f \text{ is the Jacobian of } f \text{ at time } k \tag{6}$$

4.2 Observation

The observation equation for i^{th} landmark can be written as

$$z_i = h_i(x_{k-1} \mid x_{k-1}) + w_i \tag{7}$$

where w_i is the uncertainty of observation which is temporally uncorrelated and zero-mean random noise. For point landmarks, we use both range and bearing information. For line landmarks, we only use the bearing information. The following are their measurement functions.

- Point landmarks:

$$z_i = \begin{bmatrix} r_i \\ \theta_i \end{bmatrix} = \begin{bmatrix} \sqrt{(x_i - x_r)^2 + (y_i - y_r)^2} \\ arcTan2(y_i - y_r, x_i - x_r) - \theta_r \end{bmatrix} + w_i \tag{8}$$

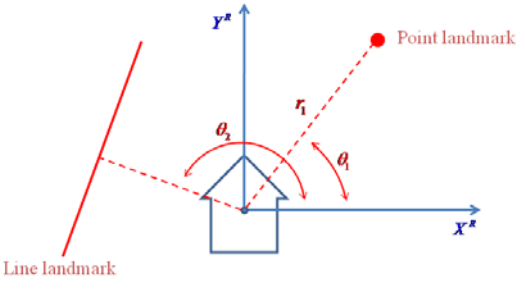

Fig. 5. Relation between landmarks and robot

- Line landmarks:

$$z_i = normal\_vector(line_i) + w_i \tag{9}$$

4.3 Data Association

For point landmarks, we use the Mahalanobis distance, which makes the distance error measurement take the correlation of the data set into account. If the Mahalanobis distance between the observed landmark z and the recorded landmark h_i is smaller than the threshold γ, we determine that z is associated with h_i.

$$v_i S_i^{-1} v_i^T < \gamma \qquad where \quad v_i = z - h_i \quad , \quad S_i = J_{h_i} P J_{h_i}^T + R \tag{10}$$

For line landmarks, we use the bearing information and the distance of the center of robot to feature line for association.

$$\left|\theta - \theta_i\right| < \gamma_\theta \qquad \left|d - d_i\right| < \gamma_d \tag{11}$$

4.4 Update

After landmark extraction and association, the measurement residual of associated landmarks can be used for EKF update. The Kalman gain K_k is computed as

$$\mathrm{K}_k = \mathrm{P}_{k|k-1} J_{hk}^T / \mathrm{S}_k \qquad where \quad J_{hk} = \frac{\partial h}{\partial \mathrm{x}_k}, \quad \mathrm{S}_k = J_{hk} \mathrm{P}_{k|k-1} J_{hk}^T + \mathrm{R}_{H_k} \tag{12}$$

The state vector and covariance are updated as

$$\begin{aligned} \mathrm{P}_{k|k} &= (\mathrm{I} - \mathrm{K}_k J_{hk}^T)\mathrm{P}_{k|k-1} \\ \hat{\mathrm{x}}_k &= \hat{\mathrm{x}}_{k|k-1} + \mathrm{K}_k v_k \qquad where \quad v_k = z_k - h_k(\hat{\mathrm{x}}_{k|k-1}) \end{aligned} \tag{13}$$

5 Path Planning for Robot Navigation

The robot navigation consists of two path planning parts. The first part is the large-scale path planning, which is similar to people choosing the appropriate path for walking along to the destination. In our system, we apply a search based method A* with big grid size. The big grid size planning has rough results but makes the plan accomplished rapidly. In the first step we do not really need a precise path because the next step deals with the obstacle avoidance. The second step, local-scale path planning, is composed of an orientation decision method and a RRT-based path planning. The path planning in this step can rapidly generate a substitute path for path smoothing or collision avoidance. The orientation decision guarantees a real-time command for reducing risks, even if the planning is not achieved in deliberation time.

5.1 Large-Scale Path Planning

In large-scale path planning, we use A* algorithm to find the path from the pre-built environment map, as shown in figure 7. A* is a best-first search using a heuristic cost function $f(x) = g(x) + h(x)$, where $g(x)$ is the cost from the starting node to the current node, and $h(x)$ is the heuristic estimation of the cost from the current node to the destination.

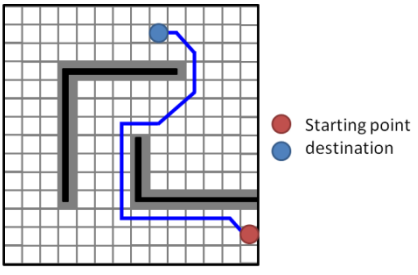

Fig. 6. Large-scale path planning by A*

5.2 Local-Scale Path Planning for Obstacle Avoidance

The local-scale path planning takes place in two situations. When the original planned path is blocked by any obstacles, re-planning applied for a collision-free path. When the original path is including extremely sharp turning angle, re-planning is applied for a smooth path.

- **RRT-based path planning.**

```
GrowTree(tree)
          1  while(x_goal != x_new)
          2    x_target = GenerateTarget();
          3    x_nearest = NearestNeighbor(x_target, tree);
          4    x_new = Extend(x_nearest, x_target);
          5    if(CollisionCheck(x_new));
          6      tree.add(x_new);
GenerateTarget()
          7  p = RandInt()%100;
          8  if(p < λ_g)
          9    return(x_goal)
          10 else
          11   return RandomPoint();
Main()
      12  tree.init(x_start);
          13  GrowTree(tree);
```

Fig. 7. The RRT Algorithm

The standard RRT algorithm is shown in Figure 7. To grow the tree, first x_{target} is randomly sampled from the configuration space by function **GenerateTarget**. To make the tree grow more efficiently and focused on x_{goal}, **GenerateTarget** returns x_{goal} with probability λ_g. Then, the **NearestNeighbor** function finds $x_{nearest}$, which is the tree node closest to x_{target}. After that, a new node is generated by **Extend** function. If the new node is free from collision, add the new node in. Else, no extension applied. These steps are repeated until x_{goal} is reached.

In our system, we grow the RRT in three dimensional space, including the 2-D location x, y and the robot orientation θ. θ is used for smooth path generation. The **NearestNeighbor** function considers (14) as the distance between x_i and x_{target}.

$$\left|\overrightarrow{x_i\ x_{\text{target}}}\right| + \omega \cdot \left(dir(\overrightarrow{x_i\ x_{\text{target}}}) - orientation(x_i)\right) \tag{14}$$

,where ω is the weighting const.

The **Extend** function also needs to consider the orientation. For the smoothness of the generated path, x_{new} can only turn θ_{th} even if the orientation difference between x_i and $\overrightarrow{x_i\ x_{\text{target}}}$ is larger than it.

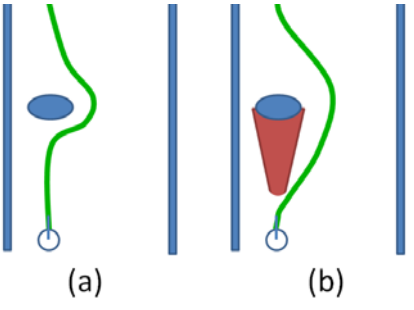

Fig. 8. Illustration of risk region

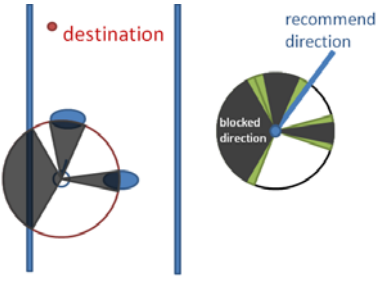

Fig. 9. Illustration of orientation decision

Furthermore, the RRT tends to generate path along the obstacle barrier, as shown in Figure 8(a). This is not a good property especially when the obstacle is moving. We set the risk region between obstacle and robot to make the planned path response to obstacle earlier. Those points in the risk region will have larger cost when connecting with the tree nodes to reduce the probability of planed path crossing through it.

- **Orientation Decision**

Although the RRT algorithm could be very fast in local-scale path planning, there are still some cases where the planning cannot be completed in real-time. Many works proposed the "anytime" version of RRT to cope with the time-limited problem. Although speeded up, still not guaranteed in real-time. In our system, we do not try to

guarantee real-time generating the path but to find the appropriate direction in real-time for robot to follow. Because of the advantage of laser range finder, we can easily get the block distance in every angle degree. If the distance is smaller than a threshold, we determine that this angle degree is blocked, as shown in Figure 9. We also consider the angle degrees near the blocked angle degrees are in risk (green regions). Then, we choose the direction closest to the destination direction from the free degrees as the recommended direction. Once the RRT planning cannot be completed in deliberated time, we use the recommended direction for improving obstacle avoidance.

6 Experimental Results

6.1 Simultaneous Localization and Mapping

In the SLAM experiment, the robot is controlled to run a closed loop in a long corridor. For the outward part (downward), localization and building map at the same time. For the return part (turning & upward), localization is applied only. Figure 10 shows the results of SLAM using different landmarks. Table 1 shows the error between the starting point and ending point. Figure 10(a) shows the result using only odometer data. Figure 10(b) shows the result using the point landmarks. The result using the line landmarks is shown in Figure 10(c). And Figure 10(d) shows the result using both point and line landmarks.

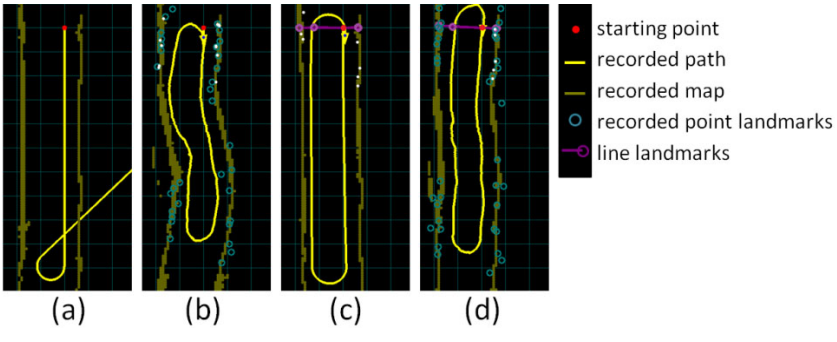

Fig. 10. Results of SLAM using different landmarks

Table 1. Error comparison

	X	Y
Ground Truth	0.0 m	0.0 m
(a)	6.23 m	-3.11 m
(b)	0.07 m	-0.41 m
(c)	0.13 m	-0.31 m
(d)	0.05m	0.00 m

From Figure 10 and Table 1, we can find that the result (a) has large error, especially when the robot is turning, the robot completely missed its orientation. Because (a) does not use any landmarks for error correction, the error is continuously accumulated. Therefore, the final estimated location has extremely large error. Result (b) uses the point landmarks for error correction and performs much better than (a). However, when the robot goes through a section with fewer point landmarks, the error of orientation becomes large and makes the built map distorted. Result (c) using the line landmarks, which are used for orientation correction, is almost perfect in orientation estimation. However, because there is no x-y lacation compensation, the straigh path is longer than the ground truth. Result (d), which is the method used in our system, can both correct the x-y location and orientation. Although there are still tiny error occuring on the recorded path, the robot can continuously compensate errors and find the location by itself.

6.2 Navigation

In the experiment of robot navigation, we first simulate our local-scale path planning method to see if it can really generate a good path for static and dynamic obstacle avoidance. Figure 11 shows the simulation environment and Table 2 shows the statistical results. The black squares in Figure 11 represent the static obstacles. The blue circles represent the dynamic obstacles, their speed and moving direction is randomly generated. As shown in Table 2, the success probability with three dynamic obstacles is higher than 90%, and the success probability with orientation decision is higher than that without orientation decision.

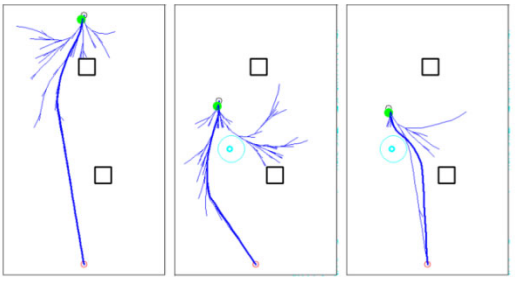

Fig. 11. Simulations of obstacle avoidance

Table 2. Success rate comparison

static obstacle number	dynamic obstacle number	Success(/100 times) (with orientation decision)	Success(/100 times) (without orientation decision)
2	1	99%	99%
2	2	96%	94%
2	3	91%	89%

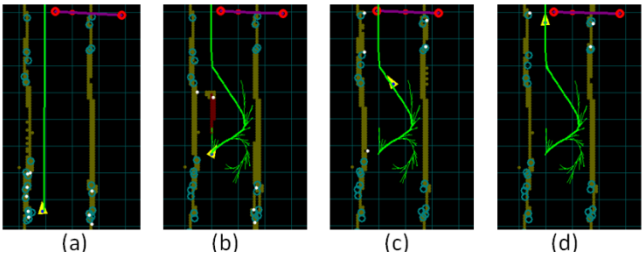

Fig. 12. On-road testing of obstacle avoidance

Beside the simulation results, we also have the on-road testing as shown in Figure 12. In Figure 12(a), the robot follows the original path planned by large-scale planning. The robot detects the obstacle and generates a substitute path for collision avoidance, as shown in (b). This on-road testing shows the robot can continuously localize itself, even if the obstacle hides some landmarks, and follow the substitute path to move to the destination.

7 Conclusion

In this paper, we propose a system using omni-directional camera and laser range finder for robot SLAM and navigation in indoor environments. We extract the point features and the line features as the landmarks. In SLAM and self-localization while navigation, the uncertainty of the odometer and observation is compensated by applying the linearized system model and odometer data to the extended Kalman filter (EKF). By the error compensation, the robot pose and the landmark feature can be well estimated. After the map has been built, robot can navigate in the environment based on it. We apply two-scale path-planning for navigation. The large-scale planning finds an appropriate path from starting point to destination. The A* with big grids is used in this step. The local-scale path-planning fills up the drawbacks of the prior step, such as dealing with the static and dynamic obstacles and smoothing the path for easier robot following. We apply an improved RRT algorithm for the path-planning in this step and use an orientation decision method to guarantee the real-time response to the detected obstacles.

Through the experiment results, we showed that the proposed system can smoothly and correctly locate itself, build the environment map and navigate in indoor environment. With the advantage of wide sensing field sensors, the self-localization still works even when there are obstacles covering some landmarks or the robot continuously changes the orientation to avoid collision.

References

1. Smith, R., Self, M., Cheeseman, P.: Estimating uncertain spatial relationships in robotics. In: Cox, I.J., Wilfong, G.T. (eds.) Autonomous Robot Vehicles, pp. 167–193. Springer, Heidelberg (1990)

2. Smith, R.C., Cheeseman, P.: On the representation and estimation of spatial uncertainty. Technical Report TR 4760 & 7239, SRI (1985)
3. Durrant-Whyte, H., Bailey, T.: Simultaneous localization and mapping: part I. IEEE Robotics & Automation Magazine 13(2), 99–110 (2006)
4. Bailey, T., Durrant-Whyte, H.: Simultaneous localization and mapping: part II. IEEE Robotics & Automation Magazine 13(3), 108–117 (2006)
5. Doucet, A., de Freitas, J.F.G., Murphy, K., Russel, S.: Rao-Blackwellized particle filtering for dynamic Bayesian networks. In: Proc. of the Conf. on Uncertainty in Artificial Intelligence (UAI), Stanford, CA, USA, pp. 176–183 (2000)
6. Civera, J., Davison, A.J., Montiel, J.: Inverse Depth Parametrization for MonocularSLAM. IEEE Trans. on Robotics 24(5), 932–945 (2008)
7. Chang, H.H., Lin, S.Y., Chen, Y.C.: SLAM for Indoor Environment Using Stereo Vision. In: Second WRI Global Congress on Intelligent Systems (2010)
8. Kuo, B.W., Chang, H.H., Lin, S.Y., Chen, Y.C., Huang, S.Y.: A Light-and-Fast SLAM Algorithm for Robots in Indoor Environments using Line Segment Map. Journal of Robotics (2011)
9. Dijkstra, E.W.: A note on two problems in connection with graphs. Numerische Mathematik 1, 269–271 (1959)
10. Hart, P., Nilsson, N., Raphael, B.: A formal basis for the heuristic determination of minimum cost paths. IEEE Transactions on Systems Science and Cybernetics 4(2), 100–107 (1968)
11. Stentz: Optimal and Efficient Path Planning for Partially-Known Environments. In: Proceedings of 1994 IEEE International Conference on Robotics and Automation, vol. 4, pp. 3310–3317 (May 1994)
12. Zilberstein, S.: Using Anytime Algorithms in Intelligent Systems. AI Magazine (Fall 1996)
13. Likhachev, M., Ferguson, D., Gordon, G., Stentz, A., Thrun, S.: Anytime Dynamic A*: An Anytime, Replanning Algorithm. In: International Conference on Automated Planning & Scheduling (2005)
14. Overmars, M.: A random approach to motion planning. Tech. rep., Utrecht University (October 1992)
15. LaValle, S.M.: Rapidly-Exploring Random Trees: A New Tool for Path Planning. Tech. Rep. 98-11, Iowa State University, Ames, IA (October 1998)
16. Ferguson, D., Stentz, A.: Anytime RRTs. In: Proceedings of the IEEE International Conference on Intelligent Robots and Systems, IROS (2006)
17. Ferguson, D., Stentz, A.: Anytime, Dynamic Planning in High-dimensional Search Spaces. In: Proceedings of the IEEE International Conference on Robotics and Automation (2007)