Journal of Zhejiang University-SCIENCE C (Computers & Electronics)
ISSN 1869-1951 (Print); ISSN 1869-196X (Online)
www.zju.edu.cn/jzus; www.springerlink.com
E-mail: jzus@zju.edu.cn

Map building for dynamic environments using grid vectors*

Wen-fei WANG, Rong XIONG†‡, Jian CHU

(*State Key Laboratory of Industrial Control Technology, Institute of Cyber-Systems and Control, Zhejiang University, Hangzhou 310027, China*)

†E-mail: rxiong@iipc.zju.edu.cn

Received July 19, 2010; Revision accepted Dec. 13, 2010; Crosschecked May 5, 2011

Abstract: This paper addresses the problem of creating a geometric map with a mobile robot in a dynamic indoor environment. To form an accurate model of the environment, we present a novel map representation called the 'grid vector', which combines each vector that represents a directed line segment with a slender occupancy grid map. A modified expectation maximization (EM) based approach is proposed to evaluate the dynamic objects and simultaneously estimate the robot path and the map of the environment. The probability of each grid vector is evaluated in the expectation step and then used to distinguish the vector into static and dynamic ones. The robot path and map are estimated in the maximization step with a graph-based simultaneous localization and mapping (SLAM) method. The representation we introduce provides advantages on making the SLAM method strictly statistic, reducing memory cost, identifying the dynamic objects, and improving the accuracy of the data associations. The SLAM algorithm we present is efficient in computation and convergence. Experiments on three different kinds of data sets show that our representation and algorithm can generate an accurate static map in a dynamic indoor environment.

Key words: Grid vector, Line, Segments, Dynamic, Simultaneous localization and mapping (SLAM), Expectation maximization (EM)

doi:10.1631/jzus.C1000255 **Document code:** A **CLC number:** TP242

1 Introduction

Learning maps of unknown environments is generally regarded as one of the fundamental problems in the pursuit of building truly autonomous mobile robots. In previous work, such a task is usually referred to as a simultaneous localization and mapping (SLAM) problem (Thrun, 2002; Frese, 2006). Under the assumption of static environments, the SLAM has been formulated and solved as a probabilistic problem in a number of different forms, such as the extended Kalman filter (EKF) (Williams *et al.*, 2001), extended information filter (EIF) (Walter *et al.*, 2007), expectation-maximization (EM) (Thrun *et al.*, 1998), particle filter (Hahnel *et al.*, 2003; Grisetti *et al.*, 2007b), and least-square error minimization techniques (Lu and Milios, 1997a; Gutmann and Konolige, 1999).

However, the real-world environments are usually not static. They contain various dynamic objects, such as people that may move freely, doors that may change the status of open or closed, and temporary furniture that may alter the location over time. The representation used for such an environment should be efficient to model both the static structures and the dynamic objects, and the corresponding estimation technique should be able to handle moving objects, by either filtering out or tracking. Otherwise, the dynamic aspects will lead to false localization and mapping.

In this paper, a new representation and its corresponding SLAM algorithm are presented for mapping in dynamic indoor environments with mobile

‡ Corresponding author

* Project supported by the National Natural Science Foundation of China (Nos. 60675049 and 61075078) and the National High-Tech Research and Development Program (863) of China (No. 2008AA04Z209)

robots. The new representation introduces the grid vector as a basic element to represent the environment, which combines each vector in the map with the occupancy grid. The vector in this paper represents a line segment with a direction to distinguish the open space and unobserved space divided by the segment. Using the probability provided by the grid vector, the associated objects in the environment can be distinguished into static and dynamic ones. A modified EM-SLAM (Thrun *et al.*, 1998) algorithm is applied to iteratively evaluate the dynamic objects and simultaneously estimate the robot path as well as the map of the environment. In the expectation-step (E-step), a statistic method is used to evaluate the probability of grid vectors considering the most likely map at that time. In the maximization-step (M-step), a graphical SLAM algorithm (Olson *et al.*, 2006; Grisetti *et al.*, 2007a) based on pose network constructed only by the expected static vectors is adopted to estimate both the robot path and the map. The two steps are carried out alternately until no further improvement can be achieved. The grid vector map we propose can be regarded as an object-level model. Separating the vectors into dynamic and static ones helps us a lot to analyze the objects in the environments. Experimental results show that our approach can filter out dynamic aspects like moving people and model special structures such as doors.

2 Related work

Landmarks (Williams *et al.*, 2001; Walter *et al.*, 2007), occupancy grid (Hahnel *et al.*, 2003; Grisetti *et al.*, 2007b), and line segments (Zhang and Ghosh, 2000; Brunskill and Roy, 2005; Connette *et al.*, 2007; Sohn and Kim, 2009) are popular representations of a map. They are widely used in generating metric maps for static environments. Recently, researchers have applied them in the mapping of dynamic environments. Most of the existing estimation techniques treat dynamic aspects as outliers, which are detected and filtered during the mapping procedure.

Bailey (2002) maintained a contemporaneous map by removing landmarks that have become obsolete due to changes in the environment. Montesano *et al.* (2005) described an algorithm for simultaneously tracking moving objects and estimating the pose of static landmarks. But these approaches require pre-defined features to model the static environments as well as moving objects into landmarks.

Occupancy grid map is the most famous representation for both static and dynamic environments owing to the probabilistic property. It benefits a list of strictly statistical methods for solving the SLAM problem or identifying a dynamic object. Yamauchi and Langley (1997) developed a technique for place learning and place recognition in a dynamic environment by decaying the occupancy probability in a grid over time. Biswas *et al.* (2002) proposed an algorithm to learn the models of the dynamic objects represented by local and object-specific occupancy grid maps. In general, grid mapping is an inefficient representation of environments, however, in terms of memory requirements.

Many researchers have focused on the segment-based map, which is efficient in view of memory requirement and data association. Moreover, line segments can be used in object-based modeling. The static walls and dynamic doors in the environment can be represented clearly (Anguelov *et al.*, 2002). But the pure segment-based map lacks probabilistic information. Some researchers constructed probabilistic models for segment parameters during segmentation by holding the errors of sensor readings (Zhang and Ghosh, 2000; Sohn and Kim, 2009). Such methods endow the segments with the probability feature, thus making the segment-based map suitable for some probabilistic underlying estimation methods such as EKF (Connette *et al.*, 2007) and particle filters (Brunskill and Roy, 2005). However, these methods are for the static environment. In a populated environment, the segment-based map is successfully applied in localization (Siegwart *et al.*, 2003) by taking advantage of its intrinsic capability to filter the range measurements that are not on a line, such as moving people. Little of the literature applies it to mapping in the dynamic environment.

We present a new map representation called the 'grid vector' in this paper, which has the advantages of both the occupancy grid and segment vectors. Each vector in the map is combined with a slender grid map. This grid map maintains the existing probability of the vector by recording the occupied probability of the locations around the vector. Using such a representation, not only probabilistic mapping algorithms can be adopted conveniently because of its grid property, but also the data association becomes easier for the vector's object-based

property. During the SLAM procedure, where an EM algorithm is used, the vectors are classified into dynamic and static ones. Dynamic vectors are then filtered out in data association and SLAM processes, so we can obtain more accurate models of the environment. Moreover, the filtered vectors provide evidence for analyzing models of the dynamic objects, and help us detect and model special structures in the environment such as doors.

3 Grid vector map

The new map representation called the 'grid vector' is introduced in this section. We present how to form a grid vector by combining a vector with a grid map, and describe the structures and properties of the new representation in detail. The grid vector mapping algorithm is presented to calculate the posterior probability over maps. A robust vector merging procedure based on grid vectors is shown in the end of this section.

3.1 Grid vector extraction

To provide probabilities in a vector map, we combine each vector with a 2D grid map (Fig. 1). The width of the grid map is very small, because only the probabilities of the grids around the vector need to be recorded. The length of the grid map is determined by the length of the vector and the resolution. As mentioned in the literature (Arras and Siegwart, 1997; Borges and Aldon, 2004; Sohn and Kim, 2009), a segment extracted with a direction to specify the observed orientation makes itself a vector. In this study, we assume the left side of a vector is an open space and the other side an unobserved space (Fig. 1). Two vectors pointing to opposite directions cannot correspond to each other.

For each range scan z_t, we can extract several grid vectors $\boldsymbol{z}_t \simeq [\mathfrak{v}_{1,t}, \mathfrak{v}_{2,t}, \ldots, \mathfrak{v}_{n,t}]$. These grid vectors compose a local map $\mathcal{V}_t$.

A grid vector in a local map is defined as

$$\mathfrak{v}_{i,t} = [[(x_{\mathrm{s}_i}, y_{\mathrm{s}_i}), (x_{\mathrm{e}_i}, y_{\mathrm{e}_i})], \mathfrak{g}_i, c_i, t], \qquad (1)$$

where $[(x_{\mathrm{s}_i}, y_{\mathrm{s}_i}), (x_{\mathrm{e}_i}, y_{\mathrm{e}_i})]$ are the ith vector's geometry parameters, $(x_{\mathrm{s}_i}, y_{\mathrm{s}_i})$ is the start point, $(x_{\mathrm{e}_i}, y_{\mathrm{e}_i})$ is the terminal point, $\mathfrak{g}_i$ is the grid map combined with the ith vector, c_i represents the dynamic property generated from the grid, and t is the time when this vector is extracted according to its observing robot pose $\boldsymbol{x}_t$.

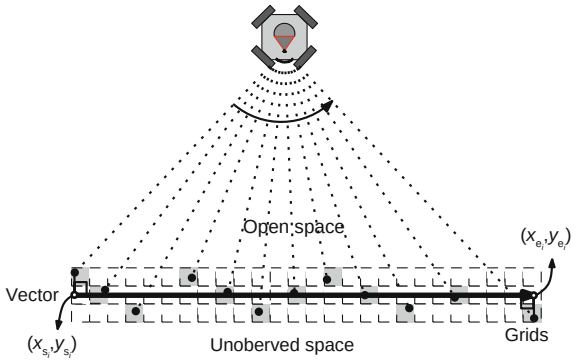


Fig. 1 An example of a vector grid. The left side is an open space and the right side is an unobserved space

In the expression of the grid vector, $\mathfrak{g}_i$ plays an important role in providing various information on vector's probabilities, including the dynamic probability for the grid map and the geometry probability for the line segment. We suppose each vector has a binary value c_i, which specifies whether a vector represents a static object or a dynamic object. Using '1' for static and '0' for dynamic, the notation $p(c_i = 1)$ or $p(c_i)$ is used to represent the dynamic probability, which is the probability that a vector is static. The geometry probability is the covariance of the vector's geometry parameters such as vector's direction or endpoints, which can be estimated by considering all the accumulated sensors contained in the grid map.

The data structure of each cell in the grid is

$$\mathfrak{e} = \{(x_{\mathrm{acc}}, y_{\mathrm{acc}}), n_{\mathrm{acc}}, (n_{\mathrm{occupy}}, n_{\mathrm{visit}})\}, \qquad (2)$$

where the parameters $(x_{\mathrm{acc}}, y_{\mathrm{acc}})$ and n_{acc} are used to record the laser beams' endpoints influencing this cell. x_{acc} and y_{acc} are the accumulations on the x- and y-axis of all the beams ended in this cell respectively, and n_{acc} records the counts. If $n_{\mathrm{acc}} > 0$, then $(x_{\mathrm{acc}}, y_{\mathrm{acc}})/n_{\mathrm{acc}}$ represents the gravity center of all the points in this cell. Otherwise, it is a blank cell that no beam ends in. The above parameters are changed only in the initialization and vector merging steps, used to represent the probability of the vector's geometry property. The last parameters $(n_{\mathrm{occupy}}, n_{\mathrm{visit}})$ represent the occupancy probability of the cell and are updated by a counting model. Considering a laser beam, if its endpoint drops in a grid cell, then the occupancy count n_{occupy} and visited count n_{visit} of this cell both grow up. Otherwise,

if the beam passes this grid, only n_{visit} increases. With a priori probability in the counts, i.e., having $n_{\text{occupy}} = 1/2$ and $n_{\text{visit}} = 1$, the occupancy probabilities are computed using the form of $n_{\text{occupy}}/n_{\text{visit}}$. The essence of this counting technique is to count for each cell how often beams end in that cell or cover but not end in it.

We use the split-and-merge method (Borges and Aldon, 2004) to extract line segments from range scans. A least-squares approach (Lu and Milios, 1997b) is adopted for line fitting. The line model in our study is defined as

$$\cos\theta \cdot x + \sin\theta \cdot y - \rho = 0, \tag{3}$$

where ρ is the normal distance from the origin to the vector, and θ is the angle of the normal.

When a line is extracted, its corresponding laser points are obtained. Calculating the foot of the perpendicular from each laser point to the line, the outermost two will be selected to be the endpoints of the line segment. As shown in Fig. 1, assuming that the left side of a vector is an open space, the start point $(x_{\text{s}_i}, y_{\text{s}_i})$ and terminal point $(x_{\text{e}_i}, y_{\text{e}_i})$ are selected according to the range scans' orientation. Supposing the segment vector always points from the start point to the terminal point, we can compute the other geometry parameters of the vector as the following:

$$\begin{aligned}
\psi_i &= \text{atan2}(y_{\text{e}_i} - y_{\text{s}_i}, x_{\text{e}_i} - x_{\text{s}_i}),\\
x_{\text{c}_i} &= (x_{\text{s}_i} + x_{\text{e}_i})/2,\\
y_{\text{c}_i} &= (y_{\text{s}_i} + y_{\text{e}_i})/2,\\
l_i &= \sqrt{(x_{\text{e}_i} - x_{\text{s}_i})^2 + (y_{\text{e}_i} - y_{\text{s}_i})^2}.
\end{aligned}$$

Here, ψ_i is the vector's direction, $(x_{\text{c}_i}, y_{\text{c}_i})$ is the gravity center of the vector, and l_i is the length.

Different from other methods computing parameters of probabilistic models during segment extraction, the way we represent the geometry probability is applying a counting model to update the parameters of each grid cell after the vector is extracted. The grid $\mathfrak{g}_i$ keeps the variants of a vector's parameters as an underlying way of storing the accumulations of sensors.

Initialization of the $\mathfrak{g}_i$ in the grid vector model is straightforward. Transforming the beam endpoints that belong to a vector into its grid map, we can compute which cell the point is in. If the point is inside the predefined 2D grid, then the corresponding cell is initialized by this point with the calculation shown in Algorithm 1.

Algorithm 1 Initialize the grid cell in a vector

Input: $(x_{\text{e}}, y_{\text{e}})$'s

1: **if** $(x_{\text{e}}, y_{\text{e}})$ drops in a cell $\mathfrak{c}$ in the grid $\mathfrak{g}$ **then**
2: $x_{\text{acc}} \Leftarrow x_{\text{acc}} + x_{\text{e}}$
3: $y_{\text{acc}} \Leftarrow y_{\text{acc}} + y_{\text{e}}$
4: $n_{\text{acc}} \Leftarrow n_{\text{acc}} + 1$
5: $n_{\text{occupy}} \Leftarrow n_{\text{occupy}} + 1$
6: $n_{\text{visit}} \Leftarrow n_{\text{visit}} + 1$
7: **end if**

3.2 Grid vector map building

To integrate local maps into a whole global map, we need to transform each local map $\mathcal{V}_t$ into global coordinates according to the corresponding robot pose $\boldsymbol{x}_t$. To keep the local map consistent, a vector merging procedure should be performed after transmitting. To distinguish grid vectors in different coordinates, we use $\mathfrak{v}_{i,t}$ to represent the vector in robot coordinates and $\mathfrak{m}_{i,t}$ in global coordinates, so we have $\mathfrak{m}_{i,t} = \boldsymbol{x}_t \oplus \mathfrak{v}_{i,t}$, where $\oplus$ is the pose compounding operation (Lu and Milios, 1997a).

Eq. (4) shows how to build a global map $\mathcal{M}$ by merging the local maps:

$$\begin{aligned}
\mathcal{M} &= \{\mathcal{M}_1, \mathcal{M}_2, \ldots, \mathcal{M}_t\}\\
&= \{\boldsymbol{x}_1 \oplus \mathcal{V}_1, \boldsymbol{x}_2 \oplus \mathcal{V}_2, \ldots, \boldsymbol{x}_t \oplus \mathcal{V}_t\}\\
&= \{\boldsymbol{x}_1 \oplus \mathfrak{v}_{1,1}, \boldsymbol{x}_1 \oplus \mathfrak{v}_{2,1}, \ldots, \boldsymbol{x}_1 \oplus \mathfrak{v}_{i,1},\\
&\quad \boldsymbol{x}_2 \oplus \mathfrak{v}_{1,2}, \ldots, \boldsymbol{x}_t \oplus \mathfrak{v}_{i,t}\}\\
&= \{\mathfrak{m}_{1,1}, \mathfrak{m}_{2,1}, \ldots, \mathfrak{m}_{i,t}\}\\
&\overset{\text{merge}}{=} \{\mathfrak{m}_1, \mathfrak{m}_2, \ldots, \mathfrak{m}_j\}.
\end{aligned} \tag{4}$$

Here, $\mathcal{M}_t = \boldsymbol{x}_t \oplus \mathcal{V}_t$ means transforming a local grid vector map extracted from $\boldsymbol{z}_t$ to global coordinates. $\mathfrak{m}_j$ is the grid vector in the map after vector merging. Compared to the former structure, the merged grid vectors do not have the time parameter t, because they have been part of the global map and no long belong to any single observation. The vector merging procedure will be presented in the end of this section.

Building a grid vector map also needs to calculate the posterior probabilities over maps given all the sensor data and the robot trajectory. In the case of known z^t and x^t, the vectors $\mathfrak{m}_{i,t}$ are independent, where z^t is the sequence of sensor measurements, $z^t = \{\boldsymbol{z}_1, \boldsymbol{z}_2, \ldots, \boldsymbol{z}_t\}$, and x^t is the robot trajectory,

$x^t = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_t\}$.

$$\begin{aligned} p(\mathcal{M}|z^t, x^t) &= \prod_{i,t} p(\mathfrak{m}_{i,t}|z^t, x^t) = \prod_{i,t} p(\mathfrak{g}_{i,t}|z^t, x^t) \\ &= \prod_{i,t} p(\mathfrak{g}'_{i,t}) = \prod_{i,t} \prod_{\mathfrak{e}_k \in \mathfrak{g}_{i,t}} p(\mathfrak{e}_k|z^t, x^t) \\ &= \prod_{i,t} \prod_{\mathfrak{e}'_k \in \mathfrak{g}'_{i,t}} p(\mathfrak{e}'_k). \end{aligned} \quad (5)$$

$\mathfrak{g}_{i,t}$ is the grid map in $\mathfrak{m}_{i,t}$; after all sensors update the probabilities of cells, it turns to be $\mathfrak{g}'_{i,t}$.

Eq. (5) shows that, to obtain the posterior probability of the map, we have to estimate all the probabilities of the grid cells in each grid vector. The beam model proposed by Thrun *et al.* (2005) and a counting model are used to update the grid's probability by sensor readings taken at different times. We can estimate the probability at any place in a grid vector map using the counting model:

$$\begin{aligned} &p(m_{x,y}|z^t, x^t) \\ &= p(m_{x,y}|\mathcal{M}, z^t, x^t) p(\mathcal{M}|z^t, x^t) \\ &\simeq p(m_{x,y}|\mathfrak{g}'_{1,1}, \mathfrak{g}'_{2,1}, \ldots, \mathfrak{g}'_{i,t}) \prod_{i,t} p(\mathfrak{g}'_{i,t}) \\ &\simeq \frac{\sum_{i,t} \sum_{\mathfrak{e}_k \in \mathfrak{g}'_{i,t}} I(\mathfrak{e}_k, x, y) \cdot (\mathfrak{e}_k \to n_{\text{occupy}})}{\sum_{i,t} \sum_{\mathfrak{e}_k \in \mathfrak{g}'_{i,t}} I(\mathfrak{e}_k, x, y) \cdot (\mathfrak{e}_k \to n_{\text{visit}})}, \end{aligned} \quad (6)$$

where

$$I(\mathfrak{e}_k, x, y) = \begin{cases} 1, & \text{if } (x, y) \in \mathfrak{e}_k, \\ 0, & \text{if } (x, y) \notin \mathfrak{e}_k, \end{cases} \quad (7)$$

and $\mathfrak{e}_k \to n_{\text{occupy}}$ and $\mathfrak{e}_k \to n_{\text{visit}}$ are the corresponding parameters in $\mathfrak{e}_k$. The essential of this computation is that it gathers all the influences of the grid vectors on the map at position (x, y). The grid vector map can be transformed easily to a global occupancy grid map, for which we need only to calculate each grid cell's probability by Eq. (6).

$$\begin{aligned} p(\mathcal{M}|z^t, x^t) &= \prod_{q=1}^{t} p(\mathcal{M}_p|z_q, \boldsymbol{x}_q) \simeq \prod_{q=1}^{t} p(\mathcal{M}_p|\mathcal{V}_q, \boldsymbol{x}_q) \\ &= \prod_{q=1}^{t} \prod_{\mathfrak{m}_{i,p} \in \mathcal{M}_p} \prod_{\mathfrak{v}_j \in \mathcal{V}_q} p(\mathfrak{m}_{i,p}|\mathfrak{v}_j, \boldsymbol{x}_q). \end{aligned} \quad (8)$$

The posterior probability over map using grid vectors can be computed by a closed-form solution. The straightforward method is updating the parameters in every grid by a counting model using all sensor lasers. Eq. (8) offers an optimized way to estimate the posterior probability with lower computation complexity. We employ a geometry method to simplify the probabilities updating by computing $p(\mathfrak{m}_{i,p}|\mathfrak{v}_j, \boldsymbol{x}_q)$.

Each grid vector in the global map forms a wipe-triangle using the corresponding estimated robot position and the vector's start and terminal points. The wipe-triangle updates the probability of the vector which is intersected with it. Fig. 2 illustrates the updating procedure. A wipe-triangle is constructed for the vector of $\mathfrak{v}_j$ with the robot position $\boldsymbol{x}_q$. $\mathfrak{m}_{i,p}$ is a vector intersected with this wipe-triangle. According to the intersection, $\mathfrak{m}_{i,p}$ can be divided into three kinds of parts denoted by a, b, c. The probability of each cell in the grid of $\mathfrak{m}_{i,p}$ is updated by changing the parameters $(n_{\text{occupy}}, n_{\text{visit}})$ as follows. For the grid cell in part c, both n_{occupy} and n_{visit} are increased. Because the cell in this part is near $\mathfrak{v}_j$, it is reasonable to assume that the corresponding laser beams construct the triangle end in the cell. The grid cell in part b is far from $\mathfrak{v}_j$ but inside the wipe-triangle, and then the corresponding laser beams can be regarded as covering but not ending in the cell, so only n_{visit} is updated. The probability of the cell in part a is not changed, because it locates outside the triangle. Obviously, this map generation procedure is more efficient with the updating style of region by region.

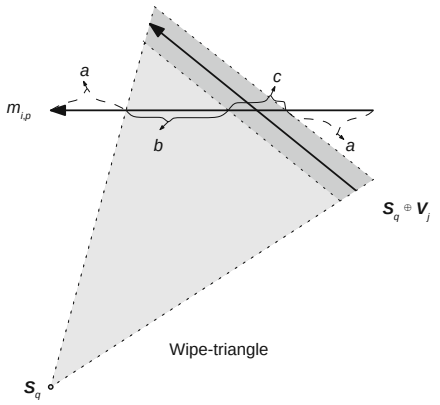


Fig. 2 Changing dynamic probabilities by geometry relationship between the wipe-triangle and intersected vector. The wipe-triangle has no effect on a, decreases the probability of part b, and increases the probability of part c

Now, we come to the problem of vector merging for grid vectors. Before merging, we have to first decide whether two vectors are corresponding to each

other. This is taken place in global coordinates. The correspondence is checked using the following three criteria:

1. The vectors' directions ψ are close.

2. The parameters θ and ϕ in the vectors' line model should be close to each other.

3. The distance between their gravities centers (x_c, y_c) is smaller than half of these two vectors' total length.

When two vectors are considered a match, they are merged in global coordinates. As illustrated in Fig. 3, a temporary grid map in the global coordinates is built for the merging process. This map is large enough to contain both grids of the two vectors. Its probability is updated according to its two-component grid maps. The parameters of each cell in the temporary grid map are computed by adding up the parameters from its corresponding cells in its components. Be aware that before adding up, the parameters (x_{acc}, y_{acc}) of each grid cell have to be transformed into the global coordinates to keep consistency with each other.

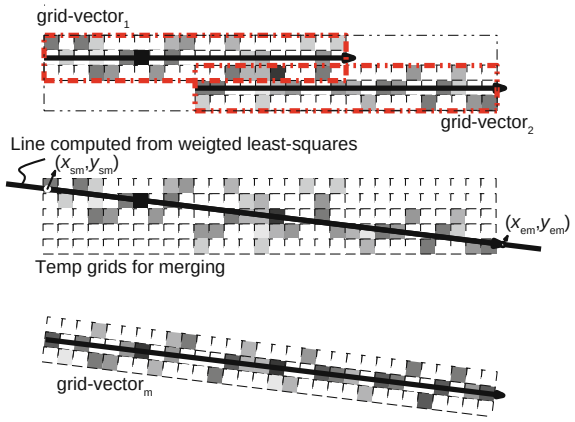


Fig. 3 Merging process for grid vectors described in Algorithm 2

From Fig. 3, we can see that, after vector merging, we use fewer grid vectors to represent the whole map, and the grids of the new vectors hold the same occupancy probability as the grid vectors before merging. That is to say, the vector merging does not change the posterior probability over the map but yields a more consistent vector map.

Most of the existing line segment algorithms take the idea to use all of the corresponding sensor endpoints to compute the segment's parameters. However, there are situations where the sensors' endpoints considered on segments are caused by dynamic objects. If we adopt the usual way in which the impact of other scans is ignored, once such a sensor's endpoint is regarded as part of a vector, it will never be changed. This will cause errors to be brought into the model for the environment. The grid vector representation we propose provides a solution to this kind of problem. As mentioned above, the grid vector keeps the existing probability distribution in its grid. After evaluating the dynamic property by all of the corresponding laser beams, we can use the occupancy probability in each cell as a weight to extend the original method of least-squares line fitting. Therefore, the negative impacts of dynamic objects can be eliminated.

The whole procedure of the vectors merging is shown in Algorithm 2.

Algorithm 2 Merging grid vectors

Input: grid-vector$_1$, grid-vector$_2$

Output: grid-vector$_m$

Form a temporary grid map
Transform two grid vectors into global coordinates
foreach grid cell in grid-vector$_1$ and grid-vector$_2$ **do**
if $n_{acc} > 0$ **then**
  Transform the points accumulation (x_{acc}, y_{acc}) into global coordinates
  $x_{acc_m} \Leftarrow x_{acc} + x_{acc_m}$
  $y_{acc_m} \Leftarrow y_{acc} + y_{acc_m}$
  $n_{acc_m} \Leftarrow n_{acc} + n_{acc_m}$
  $n_{occupy_m} \Leftarrow n_{occupy} + n_{occupy_m}$
  $n_{visit_m} \Leftarrow n_{visit} + n_{visit_m}$
end if
double ϕ, ρ
$E_{fit} = 0$
forall cells in the temporary grid **do**
if $n_{acc_m} > 0$ **then**
  $$E_{fit} += \left(\frac{x_{acc_m}}{n_{acc_m}}\cos\phi + \frac{y_{acc_m}}{n_{acc_m}}\sin\phi - \rho\right)^2 \frac{n_{occupy_m}}{n_{visit_m}}$$
end if
$(\phi^*, \rho^*) = \arg\min_{\phi,\rho} E_{fit}$
Compute new vector's endpoints $[(x_{sm}, y_{sm}), (x_{em}, y_{em})]$ from (ϕ^*, ρ^*) and the temporary grid
Form grid$_m$ for the new vector
return grid-vector$_m$ = $\{[(x_{sm}, y_{sm}), (x_{em}, y_{em})], \mathfrak{g}_m\}$

$(x_{acc_m}/n_{acc_m}, y_{acc_m}/n_{acc_m})$ is the probable central point of each grid. Thus, we do not need to store all the points that constitute individual vectors for accurate estimation. The memory requirement for the new vector does not depend on how many

range sensors or vectors are used in merging but depends only on the final length of the yielded vector. In addition, each cell participating in the computation of the least-square fitting error E_{fit} of the new vector is weighted by their occupancy probabilities $n_{\text{occupy}_\text{m}}/n_{\text{visit}_\text{m}}$. It is efficient even in the dynamic environment. A cell with a small probability will be excluded as outliers. In the static environment, the weights are simply equal to 1. To our finite knowledge, this is the only algorithm that can deal with the segments merging problem in both static and dynamic situations.

4 Learning maps in dynamic environments

The EM algorithm has been proved to be an efficient approach to the SLAM problem no matter in a static environment or a dynamic environment. However, most of the existing algorithms use the occupancy grid to represent probabilities in order to solve these problems by strict statistical derivation.

We denote the data available for mapping the form of $d^t = \{z^t, u^t\}$, where u^t is the odometry measurement, $u^t = \{u_1, u_2, \ldots, u_t\}$. In statistical terms, mapping is the problem of finding the most likely map given the data d^t. It can be formed as a maximum-likelihood (ML) estimation:

$$m^* = \arg\max_m p(m|d^t). \tag{9}$$

Because of the huge searching space, directly finding the map that globally maximizes the likelihood function is difficult. The EM algorithm solves this problem by treating it as a hidden Markov model, and performs hill climbing in the likelihood space. In static environments, Thrun *et al.* (1998) set robot's positions to be the hidden variables. The expectation over the joint log likelihood of the data d^t and the robot's path x^t is used for maximization:

$$m^{[k+1]} = \arg\max_m E_{x^t}[\log p(d^t, x^t|m)|d^t, m^{[k]}]. \tag{10}$$

In a dynamic environment, the problem is more complex, because not only the best robot's path and map have to be estimated but also the dynamic objects. This problem can be simplified by assuming that the SLAM problem in static environments is solved, which means that, if the dynamic property c_i of each grid vector is known, we can use an existing static SLAM solution to the dynamic SLAM problem with the grid vector representation. A known c_i means that we have the knowledge of whether a grid vector represents a static object or a dynamic object.

In this work, we treat the dynamic property $c^j = \{c_i, \ldots, c_j\}$ of grid vectors as the hidden variable. A modified EM algorithm is used to solve this hidden Markov problem. Supposing the map is constructed by j grid vectors, the probability $p(\mathcal{M}, x^t|d^t)$ can be written as

$$p(\mathcal{M}, x^t|d^t) = \int \cdots \int [p(\mathcal{M}, x^t|c^j, d^t) \cdot p(c^j|d^t)]\mathrm{d}c_1\mathrm{d}c_2 \ldots \mathrm{d}c_j. \tag{11}$$

Assuming that d^t and each c_i are independent given the map $\mathcal{M}$, the EM function can be formed as follows:

$$\begin{aligned}\mathcal{M}^{[k+1]} &= \arg\max_{\mathcal{M}} \log \int \cdots \int \{p(\mathcal{M}, x^t|c^j, d^t) \cdot p(c^j|d^t, \mathcal{M}^{[k]})\}\mathrm{d}c_1\mathrm{d}c_2 \ldots \mathrm{d}c_j \\ &= \arg\max_{\mathcal{M}} \int \cdots \int \{\log p(\mathcal{M}, x^t|c^j, d^t) \cdot \prod_{i+1}^{j} p(c_i|\mathcal{M}^{[k]})\}\mathrm{d}c_1\mathrm{d}c_2 \ldots \mathrm{d}c_j \qquad (12) \\ &= \underbrace{\arg\max_{\mathcal{M}} E_{c^j}[\log p(\mathcal{M}, x^t|c^j, d^t)|c^j]}_{\mathrm{M-step}} \\ &\quad + \underbrace{\sum_{i=1}^{j} E_{c_i}[\log p(c_i|\mathcal{M})|\mathcal{M}^{[k]}]}_{\mathrm{E-step}}. \qquad (13)\end{aligned}$$

Here, the right side contains the term $p(c_i|\mathcal{M})$. It is the posterior probability of the grid vector's dynamic property conditioned on the most likely map $\mathcal{M}^{[k]}$. We calculate the probability in E-step by separating vectors into dynamic and static ones.

In M-step we maximize Eq. (13) with the fixed expectations c^j obtained in E-step. The M-step is indeed another ML problem finding the most likely map $\mathcal{M}$ which yields the best interpretation of the data d^t with the knowledge of dynamic information c^j. With treating expected dynamic objects as outliers, this is a static SLAM problem, so we solve the maximization by an existing SLAM solution.

4.1 Expectation step

In E-step the expectation $p(c_i|\mathcal{M})$ for each grid vector is computed given the current map $\mathcal{M}^{[k]}$. The current map from last M-step is regarded as an approximation to the occupancy grid map built by all the sensor measurements using the beam model. The only difference is that the grid information around the vectors that we are interested in is reserved.

Because the dynamic probability c_i of a vector is calculated according to the probabilities of its accompanied grids, it is a multi-modal model. It is hard to represent and evaluate this kind of model by a parametric form. Therefore, we calculate the expectations by breaking the vector into pieces with a simple form of probability distribution.

Summing up the cells' probabilities in the y-axis, a probability distribution along this vector is generated (Fig. 4). If the probabilities are all above a threshold, this is a static vector. In contrast, it is a dynamic vector if the probabilities are all below a certain threshold. In the situations where some parts of the distribution are above and others below the threshold, we use a split-and-merge method to break this vector into static and dynamic ones.

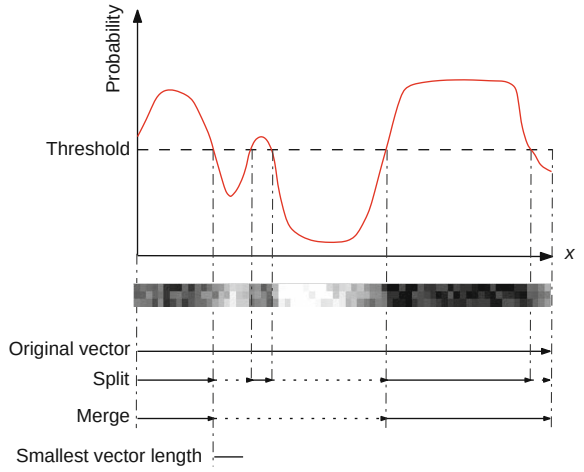


Fig. 4 Separating a vector into static (solid) and dynamic (dotted) by a split-and-merge method in the expectation step. Split at the points where the density function meets the threshold; a smallest vector is used to decide when to merge

$$p(c_i|\mathfrak{m}_i)=\begin{cases}1, & \text{if } \mathfrak{m}\in\{\mathfrak{m}_{\mathrm{s}}\},\\ 0, & \text{if } \mathfrak{m}\in\{\mathfrak{m}_{\mathrm{d}}\}.\end{cases} \tag{14}$$

$$\begin{aligned}&\sum_{i=1}^{j}E_{c_i}[\log p(c_i|\mathcal{M})|\mathcal{M}^{[k]}]\\ &\quad=\sum_{i,t}E_{c_{i,t}}[\log p(c_{i,t}|\mathfrak{m}_{i,t})]\\ &\quad=\sum_{p}\log p(c_p|\mathfrak{m}_{\mathrm{s},p})+\sum_{q}\log p(c_q|\mathfrak{m}_{\mathrm{d},q})\\ &\quad=\text{const.}\end{aligned} \tag{15}$$

After classifying the vectors $\{\mathfrak{m}_{i,t}\}$ into static $\{\mathfrak{m}_{\mathrm{s},p}\}$ and dynamic $\{\mathfrak{m}_{\mathrm{d},q}\}$, their dynamic property c_i turns to be a binary variable, which is either 1 for the static vector or 0 for the dynamic one, as shown in Eq. (14). According to Eq. (15), $\sum_{i=1}^{j}E_{c_i}[\log p(c_i|\mathcal{M})|\mathcal{M}^{[k]}]$ in Eq. (13) is now a constant.

4.2 Maximization step

The essential of M-step is a mapping problem in a dynamic environment where the dynamic objects are known, which is indeed a static SLAM problem. Unfortunately, even the static SLAM is complex and multi-modal, so we do not solve it in a closed form. The key idea of our approach is to estimate the full SLAM posterior probability by the following Bayes filter:

$$\begin{aligned}&p(\mathcal{M},x^t|d^t,c^j)\\ &=p(\mathcal{M}|x^t,d^t,c^j)\cdot p(x^t|d^t,c^j) &(16)\\ &=p(\mathcal{M}|z^t,x^t,c^j)\cdot p(x^t|z^t,u^t,c^j) &(17)\\ &\simeq p(\mathcal{M}|z^t,x^t)\cdot p(x^t|\mathcal{V}^t,u^t,c^j) &(18)\\ &=p(\mathcal{M}|z^t,x^t)\cdot p(x^t|\mathfrak{v}_{\boldsymbol{x}}^m,\mathfrak{v}_d^n,u^t) &(19)\\ &=\underbrace{\prod_{q=1}^{t}\prod_{\mathfrak{m}_{i,p}\in\mathcal{M}_p}\prod_{\mathfrak{v}_j\in\mathcal{V}_q}p(\mathfrak{m}_{i,p}|\mathfrak{v}_j,\boldsymbol{x}_q)}_{\text{mapping}}\cdot p(x^t|\mathfrak{v}_{\boldsymbol{x}}^m,u^t). &(20)\end{aligned}$$

In Eq. (16), we present the posterior probability in the same factored form as the Rao-Blackwellized particle filter. The factorization transforms the original problem into independent map generation and trajectory estimation. $p(\mathcal{M}|x^t,d^t,c^j)$ is the map generation procedure. Since c^j has no effect on the posterior probability given z^t and x^t, we can use the method introduced in Eq. (8). Eq. (19) is obtained by assuming that the E-step separates vectors into m static vectors and n dynamic vectors. As shown in Eq. (20), treating dynamic vectors as outliers,

$p(x^t|d^t, c^j)$ is completely a trajectory estimation in a static environment. Any trajectory-oriented SLAM method, such as FastSLAM and consistent pose estimation (CPE), can be used to calculate this posterior probability.

In this work, a graph-based maximum-likelihood mapping algorithm introduced by Olson *et al.* (2006) and Grisetti *et al.* (2007a) is used to estimate the most likely robot trajectory. The graph-based SLAM method treats robot poses as nodes in a graph, and estimates the most likely configuration of these nodes. Usually, the absolute poses of the poses x^t are not selected to be the estimated nodes state; instead, another state vector $\boldsymbol{S} = (s_1, s_2, \ldots, s_n)^{\mathrm{T}}$ is used, which is much easier for describing the configuration of the nodes. $\boldsymbol{S}$ can be transformed easily into global coordinates to rebuild x^t.

Assuming a Gaussian error, an edge between two nodes j and i is defined as $\mathcal{N}(\delta_{ji}, \Sigma_{ji})$, and the likelihood of the edge is its negative log probability:

$$\begin{aligned} F_{ji}(\boldsymbol{S}) :&= -\log P(f_{ji}(\boldsymbol{S})) \\ &= (f_{ji}(\boldsymbol{S}) - \delta_{ji})^{\mathrm{T}} \Sigma_{ji}^{-1} (f_{ji}(\boldsymbol{S}) - \delta_{ji}), \end{aligned} \quad (21)$$

where $f_{ji}(\boldsymbol{S})$ is a constraint function estimating the current configuration of nodes j and i. Then the graph-based SLAM algorithm uses the next ML approach to estimating the robot trajectory:

$$\boldsymbol{S}^* = \arg\max_{\boldsymbol{S}} \sum_{<j,i>\in\mathcal{E}} F_{ji}(\boldsymbol{S}). \quad (22)$$

Here, $\mathcal{E} = \{< j_1, i_1 >, < j_2, i_2 >, \ldots, < j_N, i_N >\}$ is a set of pairs of indices for which a constraint exists.

The graph-based method asks for building edges between robot poses. This building process is usually called scan registration or scan-matching. We use the method mentioned in Sohn and Kim (2008) to process vector-matching and estimate the constraints between robot poses. During vector-matching, the correspondences between vectors can be generated. In our algorithm, we store these correspondences for data association, and use them to reduce the computational complexity in EM iterations. This optimization method will be presented in Section 4.3.

A stochastic gradient descent is used to estimate the robot's path after edges are built. The solution of stochastic gradient descent makes a difficult whole ML problem into many smaller problems by optimizing the constraints individually:

$$\boldsymbol{S}^{k+1} = \boldsymbol{S}^k + \underbrace{\alpha \boldsymbol{M}^{-1} J_{ji}^{\mathrm{T}} \Sigma_{ji}^{-1} r_{ji}}_{\Delta s_{ji}}, \quad (23)$$

where r_{ji} is the residual between expected constraint δ_{ji} and its observation $f_{ji}(\boldsymbol{S})$, Σ_{ji} is the variance, J_{ji} is the Jacobian of $f_{ji}(\boldsymbol{S})$, $\boldsymbol{M}$ is the Hessian of the system, α is a learning rate, and Δs_{ji} is the result of the optimization according to constraint $< j, i >$. The total error is decreased by moving the nodes in the graph.

When the most likely robot trajectory has been estimated, we can calculate the best map by the closed-form solution presented in Section 3.

4.3 Optimization of the implementation

In practice, the constraints $\{< j, i >\}$ selected to be optimized in the graph-based SLAM are important for the computation efficiency of the EM algorithm. In our method, a data association retrieval will be performed in each M-step to control the optimization graph.

We use a tree parameterization for the nodes' state space representation, which was first introduced by Grisetti *et al.* (2007a). This parameterization rebuilds the pose graph into a tree, and can represent the topology of the environment pretty well. Searching the tree, we can obtain the minimal subgraph $\mathcal{G}_{j,i}$ that is influenced by the constraint $< j, i >$, and a subset of affected constraints $\mathcal{E}_{j,i}$. That means, if a constraint $< j, i >$ has been changed, we need only to iteratively optimize the constraints in $\mathcal{E}_{j,i}$ to make the whole graph optimal.

The selection of constraints is proven to be an efficient way to reduce the computational complexity (Grisetti *et al.*, 2008). At each M-step, we can obtain a subset of constraints $\{< j_1, i_1 >, < j_2, i_2 >, \ldots, < j_n, i_n >\}$ changed in this iteration, and then we can obtain the constraints that we have to compute this time:

$$\mathcal{E} := \mathcal{E}_{j_1,i_1} \cup \mathcal{E}_{j_2,i_2} \cup \cdots \cup \mathcal{E}_{j_n,i_n}. \quad (24)$$

The changed constraints are decided by the change of data association, which is also the change of vector correspondences here. The changes of data association have two aspects. One is the change of topology, and the other is the change of grid vector state. For each M-step, we first retrieve the pose spanning tree from the current best robot trajectory.

That is the way we evaluate the topology change of the environment. This step indeed affects the graph by adding or reducing edges. If two nodes are far apart, but their scan ranges cover each other at this time, a new edge will be built between them. If two nodes are far apart now, but there is an edge between them at the last step, the edge will be destroyed.

For the other edges in the pose graph, we evaluate the affectation of the dynamic object. We do this by adding two parameters to a normal grid vector:

$$\mathfrak{m}'_{i,t} := \{\{\mathfrak{m}_{i,t}\}, \mathcal{M}_{t_i,\mathrm{cor}}, \mathcal{M}_{t_i,\mathrm{chd}}\}. \qquad (25)$$

Here, $\mathfrak{m}_{i,t}$ is a grid vector in global coordinates according to the observation of z^t at the pose of x^t. We form the extended grid vector by adding two parameters: $\mathcal{M}_{t_i,\mathrm{cor}}$ are the grid vectors from other range scans which correspond to $\mathfrak{m}_{i,t}$, collected from the vector merging procedure; $\mathcal{M}_{t_i,\mathrm{chd}}$, the results of E-step, are the children of $\mathfrak{m}_{i,t}$. After E-step, new children will be born. Comparing the children from the last iteration, if they are different, the correspondences between this vector and others have changed. From $\mathcal{M}_{t_i,\mathrm{cor}}$ we can obtain the pairs of indices that are affected by the state change of $\mathfrak{m}_{i,t}$ (Fig. 5).

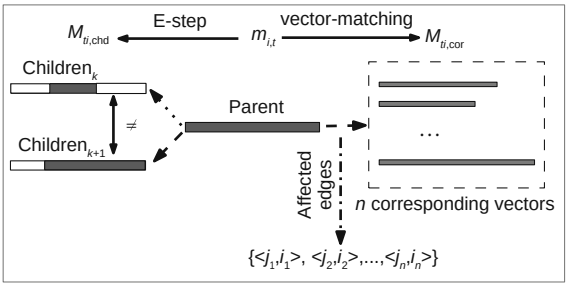


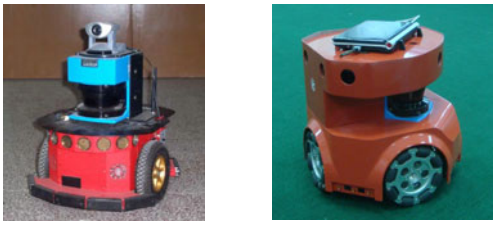
Fig. 5 Selecting affected edges from vector correspondences according to state change of the grid vector

With the knowledge above, we do not need to rebuild all the edges in the graph in the M-step, which is time expensive. If an edge contains new vector correspondences or its existing correspondences are impacted by the changed state of vectors expected in E-step, it needs to be recomputed. In other cases, the edges stay the same. The complexity of the original gradient descent method per iteration depends linearly on the number of constraints. The algorithm in Olson *et al.* (2006) has a complexity of $\mathcal{O}(MN)$, while a tree parameterization reduces it to $\mathcal{O}(M \log N)$ (Grisetti *et al.*, 2007a), where M is the number of constraints and N is the number of nodes. The M-step per iteration in our algorithm can be computed in $\mathcal{O}(l \log N)$, where l is the number of constraints generated from Eq. (24). During the iterations in the EM algorithm, l will decrease through iterations because of more and more accurate correspondences.

Recording vector correspondences has another advantage for vector merging. With known data association we can skip the correspondence checking step mentioned above. Merging the static vectors corresponding to each other, the most likely map is generated.

5 Experiments

The algorithm presented in this work has been tested on both data sets from the Internet and the data sets gathered by our two mobile robots (Fig. 6). The experimental results show the efficiency of our method in the dynamic indoor environment. Here, we present the results of three different types of data sets. After the three experiments, the method of door detection and the cost on memory and computation of our algorithm are also shown.



(a) (b)

Fig. 6 The two mobile robots used to gather the experiment data: (a) Pioneer2-DX developed by Adept MobileRobots Inc.; (b) Omnidirectional robot with a SICK LMS200 laser rangefinder, developed by our laboratory

5.1 Loop closing for a cyclic indoor environment

The first data was recorded on the cyclic floor of the USC SAL building. Fig. 7a gives the raw sensor data. Obviously, the dead-reckoning errors are significant to prevent the loop closing. Though the 'loop detection' is not specified in our algorithm (which means the constraints between the scans that should be close but do not overlap cannot be built), a topologically correct map was still obtained after

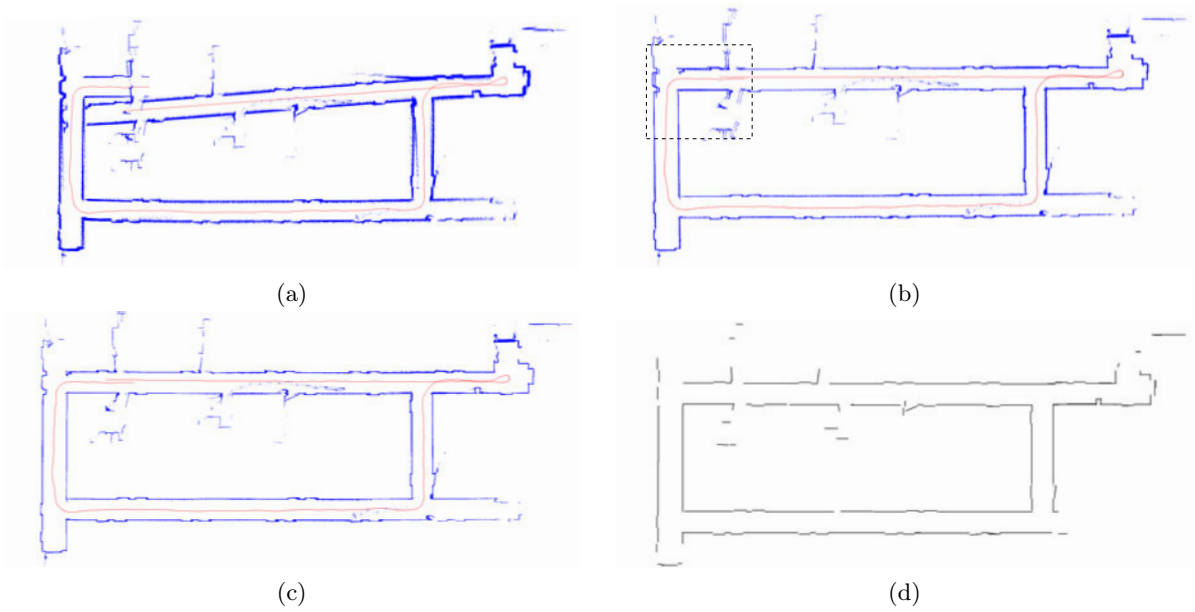


Fig. 7 (a) Map obtained in an environment of size 38 m×18 m from sensor data using raw odometry; (b) Raw sensor map after the first iteration; (c) Raw sensor map after the fourth iteration; (d) Final vector map after vector merging

the first iteration. This benefits from the method we use in M-step. As marked in Fig. 7b by the dotted rectangle, the data associations around the closed loop point can be estimated at this time due to correct identification of the overlap of the poses' scans. New data associations help to obtain more accurate maps during next iterations. Spurious sensors caused by moving people were filtered out during vector extraction. After EM procedure and vector merging, we obtained a consistent static map (Fig. 7d). This experiment result illustrates that our EM algorithm succeeds in building a map in the large cyclic as well as dynamic environment by iteratively improving the data association.

5.2 Mapping cluttered office

The second data was collected by a pioneer robot (Fig. 6a), which uses a three-wheeled motion architecture (two driving wheels and a castor wheel) in a cluttered office environment. In this environment, there are many chairs and tables inside, which causes false vector extraction during the mapping procedure. The raw sensor data and our result are shown in Fig. 8.

Take the bottom left part of the map indicated by the dotted rectangle in Figs. 8b and 8c for example. The vector extraction transforms the spurious measurement into false vectors in the map. As shown in Fig. 8d, the grid vector map records the posterior probabilities of vectors by the grids accompanied with them. After analyzing the grids' information, we separated the vectors into dynamic and static ones. Finally, the dynamic objects were picked out by merging the static vectors, and the static environment can be modeled precisely (Fig. 8e). Compared with the original occupancy map (Fig. 8f), the grid vector map has the same probability distribution around the vectors.

5.3 Mapping in a highly dynamic environment

The last experiment was carried out in a highly dynamic corridor with people walking inside and door state changing, using our four-omnidirectional mobile robot (Fig. 6b). The dead reckoning of this four-wheeled robot is a simple speed compounding with the encoder feedback of each wheel. This method usually causes large errors, especially for the translation motion accompanied with a rotation motion.

The raw odometry gathered by the omnidirectional robot has significant errors (Fig. 9a). Dynamic objects such as moving people and doors prevent the scan registration from correcting the robot pose.

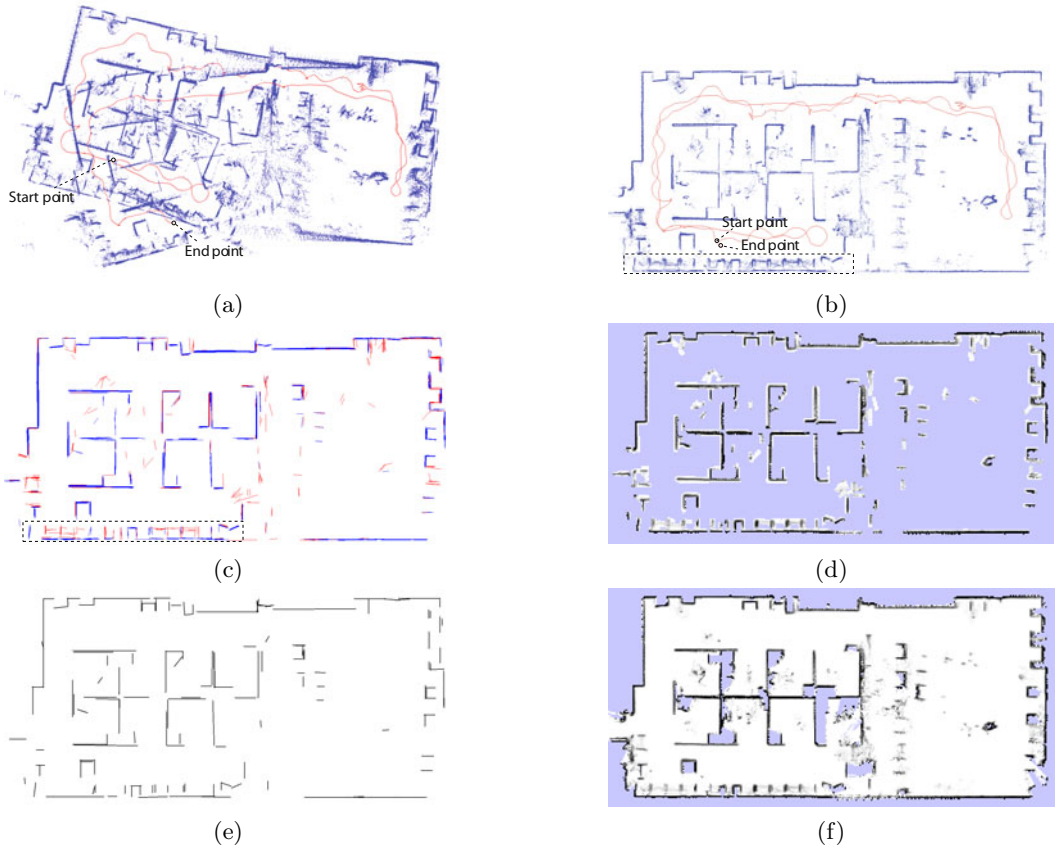


Fig. 8 (a) Map obtained in a 20 m×8 m office environment from sensor data using raw odometry; (b) Raw sensor map obtained by a corrected robot trajectory; (c) Vector representation of the grid vector map with the blue lines representing static vectors and red for dynamic; (d) Grid representation of the grid vector map; (e) Final vector map after vector merging; (f) Original occupancy map

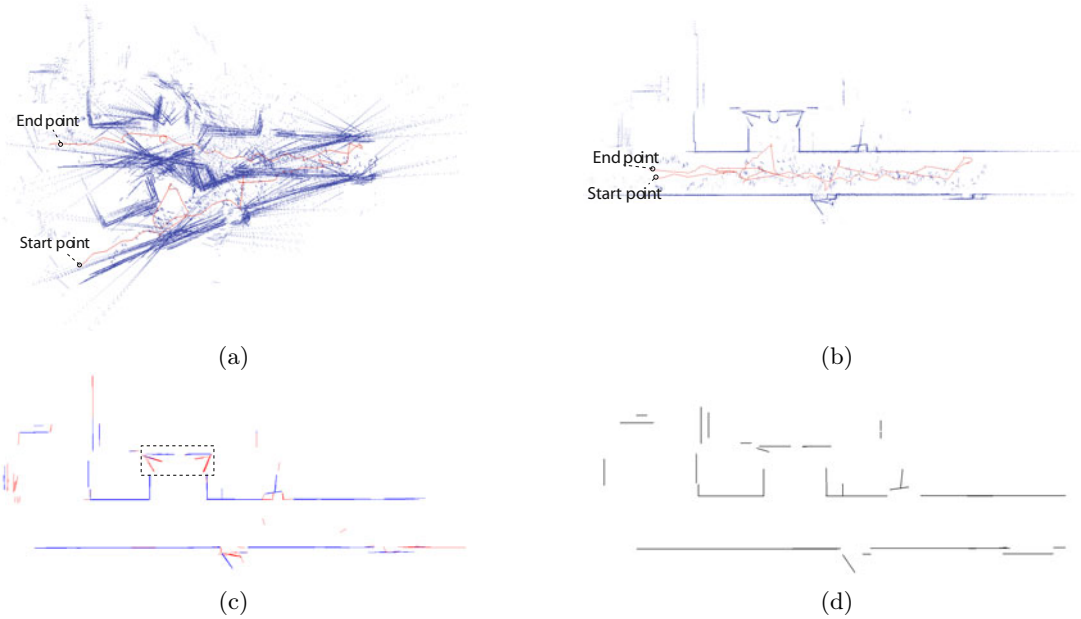


Fig. 9 (a) Raw data of a 20 m×6 m corridor gathered by the omnidirectional robot; (b) Raw measurement sensor map; (c) Vector map extracted from each range scan; (d) Final global vector map

Benefiting from vector features, the spurious laser data caused by moving people can be filtered out during vector extraction. The dynamic doors can be detected gradually in the EM procedure and excluded from data associations, so that the scan registrations can get rid of the effect from various dynamic objects and build correct constraints in the graph. Finally, the SLAM method succeeds in generating an accurate static map of the environment, even when the odometry error is large.

5.4 Door detection

For filtering dynamic objects, vectors are classified into dynamic and static ones in this work. Moreover, with this knowledge, the object-level characteristic of the grid vector provides a method for detecting doors.

Fig. 10 is part of the corridor map indicated by the dotted rectangle in Fig. 9c. Assuming that the corner is always built up by a static vector and a dynamic vector, we can obtain two corners cr_1 and cr_2 formed by the vectors $\boldsymbol{a}$, $\boldsymbol{b}$, and $\boldsymbol{c}$. Each corner has a vertex and two vectors as the borders. The static vector in the corner is probably the wall, and the dynamic one is considered as a door candidate. As shown in the figure, if two corners share the same static vector, while their vertices are close to each other, and their dynamic vectors have the same length, the vertex can be regarded as the hinge of a door, and the dynamic vectors represent different open states of the door.

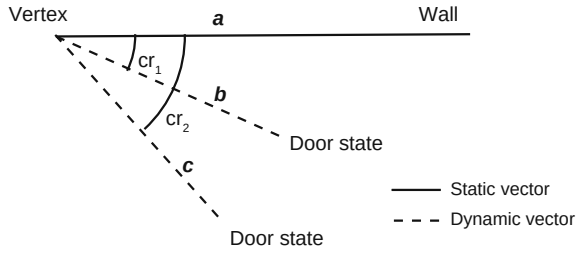


Fig. 10 Door estimation

This door estimation is easy to apply under the assumption that doors move during data collection. If a door never changes its state, it will be regarded as part of the static environment.

5.5 Memory and computation cost

Comparing Figs. 8d and 8e, the grid vector map can keep the same probabilities at the place around vectors as the original occupancy grid map. By ignoring the probability distribution apart from vectors, the grid vector map can save a lot of memory. The difference is that the size of the occupancy grid map is directly related to the area of the environment, but the grid vector map is influenced only by the total length of the vectors. We record the memory requirements between the grid map and the grid vector map for all the three data sets (Table 1). The average memory requirement of the grid map is beyond 20 times that of the grid vector map.

Table 1 Comparison of the memory requirement between the grid map and the grid vector map*

Parameter	Value		
	Fig. 7	Fig. 8	Fig. 9
Space size (m^2)	678.56	174.94	122.68
Vector length (m)	194.73	93.50	47.04
Cell number for the grid map, n_g	67 856	17 494	12 268
Cell number for the grid vector map, n_v	1947	935	471
n_g/n_v	34.85	18.71	26.05

* The grid resolution is 10 cm in all three representations

As mentioned above, our EM algorithm iteratively generates better maps by obtaining increasingly precise data association. Another aspect of this problem is the improvement of the constraints in the graph. The EM procedure converges when no constraints need to be changed. Fig. 11 records the percentage of the constraints changed in different iterations. Normally, our method needs only four or five iterations to obtain an accurate map. Thus, it has a fast convergence.

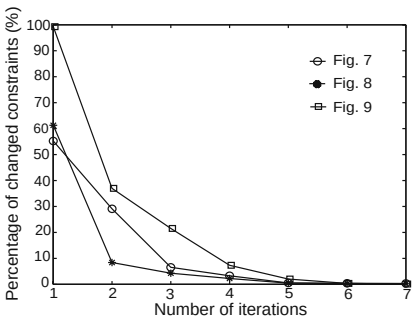


Fig. 11 Convergency of the EM algorithm

6 Conclusions

A new map representation called the 'grid vector' and an extended EM-SLAM algorithm are presented in this paper. The experiments illustrate that our method is suitable for mapping in dynamic indoor environments. It is efficient to distinguish dynamic objects and generate accurate models of the environment.

The main contributions of this paper are as follows:

1. The grid vector representation proposed provides advantages not only on filtering moving people and reducing memory cost, but also on the data association and the ability to model dynamic objects.

2. The SLAM algorithm presented takes full advantage of the vector's object-level characteristic and the grid's probability feature. It improves the computation efficiency and optimizes the convergence rate of the graph-based SLAM.

3. The SLAM algorithm presented can not only correct the robot pose, but also estimate dynamic objects according to recent observations. Thus, it can be used as a patch scan registration algorithm to replace the original scan registration step for applying other SLAM methods such as particle filters in the dynamic environment.

In the future we will extend our algorithm to sequential mode, which allows a robot to incrementally estimate the dynamic objects and the most likely map while walking through the environment.

Acknowledgements

The authors thank Andrew HOWARD for providing the data set on the website Radish http://radish.sourceforge.net/.

References

Anguelov, D., Biswas, R., Koller, D., Limketkai, B., Sanner, S., Thrun, S., 2002. Learning Hierarchical Object Maps of Non-stationary Environments with Mobile Robots. Proc. Conf. on Uncertainty in Artificial Intelligence, p.10-17.

Arras, K.O., Siegwart, R.Y., 1997. Feature extraction and scene interpretation for map-based navigation and map building. *SPIE*, **3210**:42-53. [doi:10.1117/12.299565]

Bailey, T., 2002. Mobile Robot Localisation and Mapping in Extensive Outdoor Environments. PhD Thesis, Australian Center for Field Robotics, University of Sydney, Sydney, Australia.

Biswas, R., Limketkai, B., Sanner, S., Thrun, S., 2002. Towards Object Mapping in Non-stationary Environments with Mobile Robots. IEEE/RSJ Int. Conf. on Intelligent Robots and Systems, **1**:1014-1019. [doi:10.1109/IRDS.2002.1041523]

Borges, G.A., Aldon, M.J., 2004. Line extraction in 2D range images for mobile robotics. *J. Intell. Robot. Syst.*, **40**(3):267-297. [doi:10.1023/B:JINT.0000038945.55712.65]

Brunskill, E., Roy, N., 2005. SLAM Using Incremental Probabilistic PCA and Dimensionality Reduction. Proc. IEEE Int. Conf. on Robotics and Automation, p.342-347. [doi:10.1109/ROBOT.2005.1570142]

Connette, C.P., Meister, O., Hägele, M., Trommer, G.F., 2007. Decomposition of Line Segments into Corner and Statistical Grown Line Features in an EKF-SLAM Framework. IEEE/RSJ Int. Conf. on Intelligent Robots and Systems, p.3884-3891. [doi:10.1109/IROS.2007.4399404]

Frese, U., 2006. A discussion of simultaneous localization and mapping. *Auton. Robots*, **20**(1):25-42. [doi:10.1007/s10514-006-5735-x]

Grisetti, G., Stachniss, C., Grzonka, S., Burgard, W., 2007a. A Tree Parameterization for Efficiently Computing Maximum Likelihood Maps Using Gradient Descent. Proc. Robotics: Science and Systems.

Grisetti, G., Tipaldi, G.D., Stachniss, C., Burgard, W., Nardi, D., 2007b. Fast and accurate SLAM with Rao-Blackwellized particle filters. *Robot. Auton. Syst.*, **55**(1):30-38. [doi:10.1016/j.robot.2006.06.007]

Grisetti, G., Lodi Rizzini, D., Stachniss, C., Olson, E., Burgard, W., 2008. Online Constraint Network Optimization for Efficient Maximum Likelihood Map Learning. Proc. IEEE Int. Conf. on Robotics and Automation, **1**:1880-1885. [doi:10.1109/ROBOT.2008.4543481]

Gutmann, J.S., Konolige, K., 1999. Incremental Mapping of Large Cyclic Environments. Proc. IEEE Int. Symp. on Computational Intelligence in Robotics and Automation, p.318-325. [doi:10.1109/CIRA.1999.810068]

Hahnel, D., Burgard, W., Fox, D., Thrun, S., 2003. An Efficient Fastslam Algorithm for Generating Maps of Large-Scale Cyclic Environments from Raw Laser Range Measurements. Proc. IEEE/RSJ Int. Conf. on Intelligent Robots and Systems, **1**:206-211. [doi:10.1109/IROS.2003.1250629]

Lu, F., Milios, E., 1997a. Globally consistent range scan alignment for environment mapping. *Auton. Robots*, **4**(4):333-349. [doi:10.1023/A:1008854305733]

Lu, F., Milios, E., 1997b. Robot pose estimation in unknown environments by matching 2D range scans. *J. Intell. Robot. Syst.*, **18**(3):249-275. [doi:10.1023/A:1007957421070]

Montesano, L., Minguez, J., Montano, L., 2005. Modeling the Static and the Dynamic Parts of the Environment to Improve Sensor-Based Navigation. Proc. IEEE Int. Conf. on Robotics and Automation, p.4556-4562. [doi:10.1109/ROBOT.2005.1570822]

Olson, E., Leonard, J., Teller, S., 2006. Fast Iterative Alignment of Pose Graphs with Poor Initial Estimates. Proc. IEEE Int. Conf. on Robotics and Automation, p.2262-2269. [doi:10.1109/ROBOT.2006.1642040]

Siegwart, R., Arras, K.O., Bouabdallah, S., Burnier, D., Froidevaux, G., Greppin, X., Jensen, B., Lorotte, A., Mayor, L., Meisser, M., *et al.*, 2003. Robox at Expo.02:

a large-scale installation of personal robots. *Robot. Auton. Syst.*, **42**(3-4):203-222. [doi:10.1016/S0921-8890(02)00376-7]

Sohn, H.J., Kim, B.K., 2008. An efficient localization algorithm based on vector matching for mobile robots using laser range finders. *J. Intell. Robot. Syst.*, **51**(4):461-488. [doi:10.1007/s10846-007-9194-1]

Sohn, H.J., Kim, B.K., 2009. VecSLAM: an efficient vector-based SLAM algorithm for indoor environments. *J. Intell. Robot. Syst.*, **56**(3):301-318. [doi:10.1007/s10846-009-9313-2]

Thrun, S., 2002. Robotic Mapping: a Survey. *In*: Lakemeyer, G., Nebel, B. (Eds.), Exploring Artificial Intelligence in the New Millenium. Morgan Kaufmann Publishers Inc., San Francisco, p.1-35.

Thrun, S., Burgard, W., Fox, D., 1998. A probabilistic approach to concurrent mapping and localization for mobile robots. *Mach. Learn.*, **31**(1/3):29-53. [doi:10.1023/A:1007436523611]

Thrun, S., Burgard, W., Fox, D., 2005. Probabilistic Robotics. MIT Press, Cambridge, p.153-169.

Walter, M.R., Eustice, R.M., Leonard, J.J., 2007. Exactly sparse extended information filters for feature-based SLAM. *Int. J. Robot. Res.*, **26**(4):335-359. [doi:10.1177/0278364906075026]

Williams, S., Dissanayake, G., Durrant-Whyte, H., 2001. Towards terrain-aided navigation for underwater robotics. *Adv. Robot.*, **15**(5):533-549. [doi:10.1163/156855301317033559]

Yamauchi, B., Langley, P., 1997. Place recognition in dynamic environments. *J. Robot. Syst.*, **14**(2):107-120.

Zhang, L., Ghosh, B.K., 2000. Line Segment Based Map Building and Localization Using 2D Laser Rangefinder. Proc. IEEE Int. Conf. on Robotics and Automation, **3**:2538-2543. [doi:10.1109/ROBOT.2000.846410]