MOBILE ROBOTS NAVIGATION, MAPPING & LOCALIZATION: PART II

Lee Gim Hee*
CML
DSO National Laboratories
20 Science Park Drive
Singapore 118230
Tel: (+65) 6796 8427
Email: lgimhee@dso.org.sg

Marcelo H. Ang Jr.
Department of Mechanical Engineering
National University of Singapore
9 Engineering Drive 1
Singapore 117576
Tel: (+65) 6516 2555
Fax: (+65) 6779 1459
Email: mpeangh@nus.edu.sg

INTRODUCTION

In addition to the capability to navigate from a point of origin to a given goal and avoiding all static and dynamic obstacles, a mobile robot must posses another two competencies: *map building* and *localization* in order to be useful.

A mobile robot acquires information of its environment via the process of map building. Map building for mobile robots are commonly divided into *occupancy grid* and *topological* maps. *Occupancy-grid* maps seek to represent the geometric properties of the environment. *Occupancy-grid* mapping was first suggested by Elfes in 1987 and the idea was published in his Ph.D. thesis (A. Elfes, 1989) in 1989. *Topological* mapping was first introduced in 1985 as an alternative to the *occupancy-grid* mapping by R. Chatila and J.-P. Laumond (R. Chatila, & J.-P. Laumond, 1985). *Topological* maps describe the connectivity of different locations in the environment.

The pose of a mobile robot must be known at all times for it to navigation and build a map accurately. This is the problem of localization and it was first described in the late 1980's by R. Smith et al (R. Smith et al, 1980). Some key algorithms for map building and localization will be discussed in this article.

BACKGROUND

Map building is the process of acquiring information of the environment via sensory data and representing the acquired information in a format that is comprehensible to the robot. The acquired map of the environment can be used by the robot to improve its performance in navigation.

Localization is the process of finding the pose of the robot in the environment. It is perhaps the most important competency that a mobile robot must possess. This is because the robot must know its pose in the environment before it can plan its path to the goal or follow a planned path towards the goal.

In this article, two key algorithms for map building: *occupancy-grid* and *topological* mapping are discussed. The *occupancy grid* and *topological* maps are two different methodologies to represent the environment in a robot's memory. Two key localization methods: Localization with *Kalman filter* and *particle filter* are also reviewed.

MAP BUILDING

As seen from the *integrated algorithm* from part I of the article, a mobile robot must be able to acquire maps of an unknown environment to achieve higher level of autonomy. Map building is the process where sensory information of the surrounding is made comprehensive to a mobile robot. In this section, two key approaches for map building: *occupancy-grid* and *topological* mapping are discussed.

Occupancy-Grid Maps

Occupancy-grid maps (H.P. Moravec, 1988; H.P. Moravec et al, 1989; A. Elfes, 1987, A. Elfes, 1989; S. Thrun et al, 2005) represent the environment as a tessellation of grid cells. Each of the grid cells corresponds to an area in the physical environment and holds an occupancy value which indicates the probability of whether the cell is occupied or free. The occupancy value of the i^{th} grid cell at current time t will be denoted by $p_{t,i}$. Note that $p_{t,i}$ must be within the range of 0 to 1 following the axioms

of probability. $p_{t,i}$ = [0,0.5) indicates the confidence level of a cell being empty where 0 indicates absolute certainty that the cell is empty. $p_{t,i}$ = (0.5,1] indicates the confidence level of a cell being occupied where 1 indicates absolute certainty that the cell is occupied. $p_{t,i}$ = 0.5 indicates that the cell is an unexplored area.

A robot does not have any knowledge of the world when it was first placed in an unknown environment. It is therefore intuitive to set $p_{t,i}$ = 0.5 for all i at time t = 0. The map is updated via the *log odds* (S. Thrun et al, 2005) representation of occupancy. The advantage of *log odds* representation is that it can avoid numerical instabilities for probability near 0 or 1. The i^{th} grid cell that intercepts the sensor line of sight is updated according to

$$l_{t,i} = l_{t-1,i} + l_{sensor} \tag{1}$$

where $l_{t-1,i}$ is the *log odds* computed from the occupancy value of the cell at *t-1*.

$$l_{t-1,i} = \log \frac{p_{t-1,i}}{1 - p_{t-1,i}} \tag{2}$$

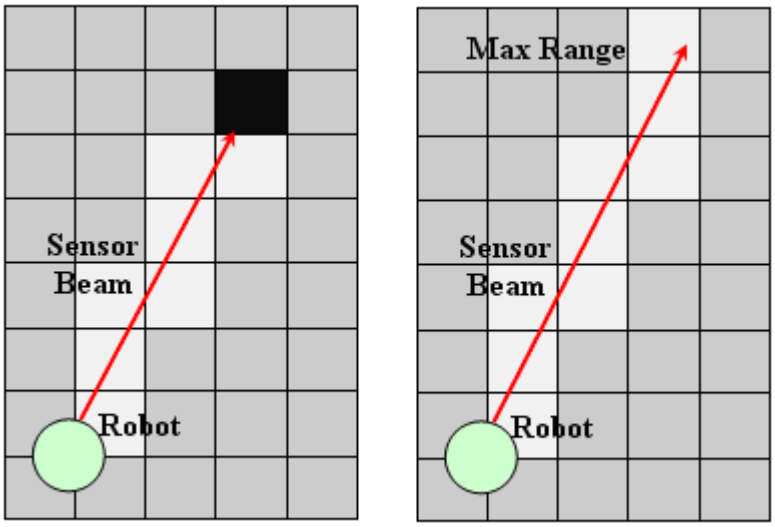
$l_{sensor} = l_{\text{occ}}$ if the cell corresponds to the sensor measurement and $l_{sensor} = l_{free}$ if the range to the cell is shorter than the sensor measurement. The other cells in the map remain unchanged.

Figure 1(a) illustrates the update process for the map. The cell that corresponds to the sensor measurement is shaded black and all the cells that intercept the sensor measurement beam are shaded white. Figure 1(b) shows a case where the sensor measurement equals to maximum sensor range and $l_{sensor} = l_{free}$ for all cells that intercepts the sensor beam. This is because it is assumed that no obstacle is detected if

the sensor measurement equals to maximum sensor range. l_{occ} and l_{free} are computed from

$$l_{occ} = \log \frac{p_{occ}}{1 - p_{occ}} \qquad \text{and} \qquad l_{free} = \log \frac{p_{free}}{1 - p_{free}} \tag{3}$$

where p_{occ} and p_{free} denote the probabilities of the sensor measurement correctly deducing whether a grid cell is occupied or empty. The two probabilities must add up to 1 and their values depend on the accuracy of the sensor. p_{occ} and p_{free} will have values closer to 1 and 0 for an accurate sensor. The values of p_{occ} and p_{free} have to be determined experimentally and remain constant in the map building process.



(a) **(b)**

Fig. 1. Updating an occupancy grid map (a) when an obstacle is detected (b) when a maximum range measurement is detected, i.e. it is assumed that in this case no obstacle is detected

The occupancy value of a grid cell is easily recovered from

$$p_{t,i} = 1 - \frac{1}{1 + \exp\{l_{t,i}\}} \tag{4}$$

Figure 2 shows an occupancy grid map of the corridor along block EA level 3 in the Faculty of Engineering of the National University of Singapore (NUS) acquired with

a laser range finder. The black regions denote obstacles, white regions denote free space and grey regions denote unexplored areas.

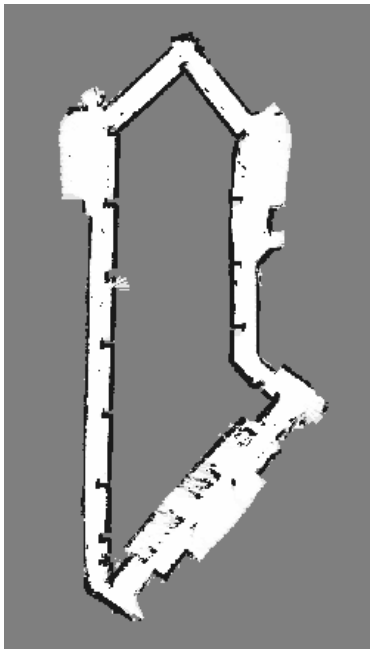


Fig. 2. Occupancy grid map of the corridor along block EA level 3 in the Faculty of Engineering of the National University of Singapore (NUS).

Topological Maps

Unlike the *occupancy grid* maps, topological maps (D. Kortenkamp et al, 1994; H. Choset, 1996; H. Choset et al, 1996) do not attempt to represent the geometric information of the environment. Instead, *topological* maps represent the environments as *graphs*. An example of the topological map is shown in Figure 3. List of significant features such as walls, corners, doors or corridors are represented as nodes m_i and connectivity between adjacent features is represented as edges u_{jk}. In many *topological* maps, distances between adjacent features are also represented by the edges connecting the nodes. The success of the *topological* maps depends greatly on the efficiency in features extraction. Examples of feature extraction algorithms can be found in (Martin David Adams, 1999; Sen Zhang et al, 2003; Jodo Xavier et al, 2005).

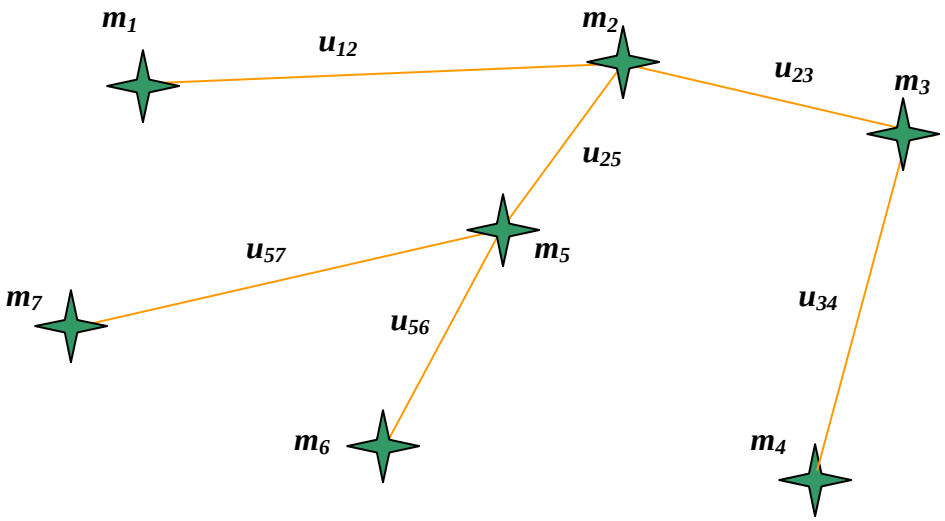


Fig. 3. Example of a topological map. The features are represented as nodes m_i. The connectivity and distance between features are represented as edges u_{jk}.

Topological maps are better choice for mapping if memory space is a major concern. This is because less memory is required to store the nodes as compared to the large number of grid cells in occupancy grid maps. The advantage of less memory consumption for the *topological* map however comes with the tradeoff of being less accurate. This is because some important information such as precise location of the free spaces in the environment may not be represented in the maps. The limited accuracy of *topological* maps thus restricts the robot's capability for fast and safe navigation.

LOCALIZATION

Most mobile robots localize their pose x_t with respect to a given map based on odometry readings. Unfortunately, wheel slippages and drifts cause incremental localization errors (J. Borenstein et al, 1995; J. Borenstein et al, 1996). These errors cause the mobile robot to lose track of its own pose and hence losing the ability to navigate autonomously from one given point in the map to another. The solution to

the localization problem is to make use of information of the environment from additional sensors. Examples of sensors used are laser range finder and sonar sensor that measure the distance between the robot and the nearest obstacles in the environment. The *extended Kalman filter* (EKF) and *particle filter* are two localization algorithms that use odometry and additional sensory data of the environment to localize a mobile robot. Both algorithms are probabilistic methods that allow uncertainties from the robot pose estimate and sensor readings to be accounted for in a principled way.

Localization with Extended Kalman Filter

EKF (John J. Leonard et al, 1991; A.Kelly, 1994; G. Welch et al, 1995; Martin David Adams, 1999; S. Thrun et al, 2005) is perhaps the most established algorithm for localization of mobile robots because of its robustness and efficiency. The EKF is a recursive algorithm for estimating the pose of the robot with noisy sensor readings. A key feature of the EKF is that it maintains a posterior belief $bel(x_t)$ of the pose estimate, which follows a *Gaussian distribution*, represented by a mean x_t and covariance P_t. The mean x_t represents the most likely pose of the robot at time *t* and covariance P_t represents the error covariance of this estimate. The EKF consists of two steps: the prediction and update steps. In the prediction step, the predicted belief $\overline{bel}(x_t)$ is first computed using a motion model which describes the state dynamics of the robot. $\overline{bel}(x_t)$ is subsequently transformed into $bel(x_t)$ by incorporating the sensor measurements in the update step.

As mentioned above, the predicted belief $\overline{bel}(x_t)$, which is represented by the predicted mean $\bar{x}_t$ and covariance $\overline{P}_t$, is computed from the prediction step given by

$$\bar{x}_t = f(x_{t-1}, u_t) \quad (5)$$

$$\overline{P}_t = F_t P_{t-1} F_t^T + Q_t \quad (6)$$

where $f(.)$ is the motion model of the mobile robot, F is the Jacobian of $f(.)$ evaluated at x_{t-1}, Q_t is the covariance of the motion model and u_t is the control data of the robot.

$\overline{bel}(x_t)$ is subsequently transformed into $bel(x_t)$ by incorporating the sensor measurement z_t into the update step of the EKF shown in Equations 7, 8 and 9.

$$K_t = \overline{P}_t H_t^T (H_t \overline{P}_t H_t^T + R_t)^{-1} \quad (7)$$

$$x_t = \bar{x}_t + K_t (z_t - h(\bar{x}_t, m)) \quad (8)$$

$$P_t = (I - K_t H_t) \overline{P}_t \quad (9)$$

K_t, computed in Equation 7, is called the *Kalman gain*. It specifies the degree to which z_t should be incorporated into the new pose estimate. Equation 8 computes x_t by adjusting it in proportion to K_t and the deviation of the z_t with the predicted measurement $h(\bar{x}_t, m)$. It is important to note that the sensor measurement $z_t = [z_t^1 \quad z_t^2 \quad ...]^T$ refers to coordinates of a set of observed landmarks instead of the raw sensor readings and the sensor measurement model $h(.)$ gives the predicted measurement from the given topological map *m* and $\bar{x}_t$. H_t is the Jacobian of $h(.)$ evaluated at x_{t-1}. Finally, the covariance P_t of the posterior belief $bel(x_t)$ is computed in Equation 9 by adjusting for the information gain resulting from the sensor measurements.

Localization with Particle Filter

In the recent years, there is an increasing interest in the use of particle filter (S. Thrun et al, 2001; C. Kwok et al, 2002; D. Fox et al, 2003; Ioannis M. Rekleitis, 2004; S. Thrun et al, 2005) over EKF for robot localization. This increased interest is likely due to four reasons. First, raw sensor measurements of the environment are used in particle filter localization where the EKF localization requires feature extraction. Second, the particle filter is more robust because unlike the EKF, it does not assume Gaussian distribution for the posterior belief $bel(x_t)$. Third, the particle filter is able to recover from localization failure. Localization failure occurs if the robot suddenly loses track of its pose during the localization process. Localization failure is also known as the kidnapped problem. Fourth, unlike the EKF there is no need to derive complicated Jacobians for the particle filter.

The intuition behind the particle filter is to represent the posterior belief $bel(x_t)$ by a finite sample set of *M* weighted particles. This sample set is drawn according to $bel(x_t)$. The particles set is denoted by

$$\xi_t = \chi_t^{[1]}, \chi_t^{[2]}, ..., \chi_t^{[M]} \tag{10}$$

where $\chi_t^{[m]} = [x_t^{[m]} \quad w_t^{[m]}]^T$ denotes the m^{th} particle. Here, $x_t^{[m]}$ is a random variable that represents a hypothesized state and $w_t^{[m]}$ is a non-negative value called the importance factor which represents the weight of each particle. Similar to the EKF, the particle filter consists of the prediction and update steps. In the prediction step, samples of the particles are drawn from a motion model of the robot to represent the predicted belief $\overline{bel}(x_t)$. The particles are then weighted according to the sensor measurements in the update step. Finally, $\overline{bel}(x_t)$ is transformed into the posterior

belief $bel(x_t)$ by resampling the particles according to their weights.

1. $\overline{\xi}_t = \xi_t = \varnothing$;
2. for m = 1 to M do
3. generate random sample of $x_t^{[m]}$ from $p(x_t \mid u_t, x_{t-1}^{[m]})$;
4. $w_t^{[m]} = p(z_t \mid x_t^{[m]}, m)$;
5. $\overline{\chi}_t^{[m]} = [x_t^{[m]} \quad w_t^{[m]}]^T$;
6. end;
7. for m = 1 to M do
8. draw $\chi_t^{[m]}$ from $\overline{\xi}_t$ with probability proportional to $w_t^{[1]}$, $w_t^{[2]}$,...., $w_t^{[M]}$;
9. end;

Table 1: Pseudo algorithm for mobile robot localization with particle filter

Table 1 shows an iteration of the recursive particle filter algorithm for localization. The inputs to the particle filter are the set of particles representing the previous state belief ξ_{t-1}, the most recent control actions u_t and measurement data z_t. Line 3 is the prediction step that generates the hypothetical state $x_t^{[m]}$ by sampling from the motion model $p(x_t \mid u_t, x_{t-1}^{[m]})$ of the robot. The set of particles obtained after *M* iterations represents $\overline{bel}(x_t)$. Line 4 computes $w_t^{[m]}$ from the sensor measurement model. The importance factor accounts for the mismatch between $\overline{bel}(x_t)$ and $bel(x_t)$. Finally, the resampling process from line 7 to 9 draws with replacement *M* particles from the temporary set $\overline{\xi}_t$ with a probability proportional to the importance factors. The distribution of $\overline{bel}(x_t)$ is transformed into $bel(x_t)$ by incorporating the importance factors in the resampling process.

Figure 4(a) to (d) shows an implementation result of a robot localizing itself in a corridor. The particle set is initialized to the initial known pose of the robot show in Figure 4(a). The particles are initialized uniformly within a circle with radii 100mm

and the initial position of the robot is taken as the center. The orientation of the particles is also initialized uniformly within $\pm 5^{o}$ to the initial orientation of the robot. This is to eliminate possible errors in estimating the initial pose of the robot. Figure 4(b) to 4(d) show that the error from the odometry grows as the robot travels a greater distance. The robot thinks that it is traveling in occupied space if it relied solely on the odometry readings and this is obviously wrong. It is apparent that the particle filter gives a more reasonable pose estimate because the robot is always moving within the free space.

It was mentioned earlier that the particle filter is able to recover from localization failure. An example of localization failure is when the robot is pushed by human resulting in a mismatch between the true and estimated pose of the robot. Fortunately, the problem can be easily solved by observing the total weights of the filter after each iteration. Localization failures will cause sharp drops in the total weights of the particles. The particles are re-initialized uniformly in the free space after detecting a sharp drop in the total weights of the particles. The particles will eventually converge to the true pose of the robot.

The particle filter is a powerful algorithm in solving the localization problem. However, it must be noted that the number of particles used to represent beliefs is an important parameter for efficiency of the particle filter in recovering from localization failures. A large size of particles is necessary to recover from localization failures in large environments and in many cases the maximum number particles is restricted by the available computing resources. This problem is also known as the *curse of dimensionality*.

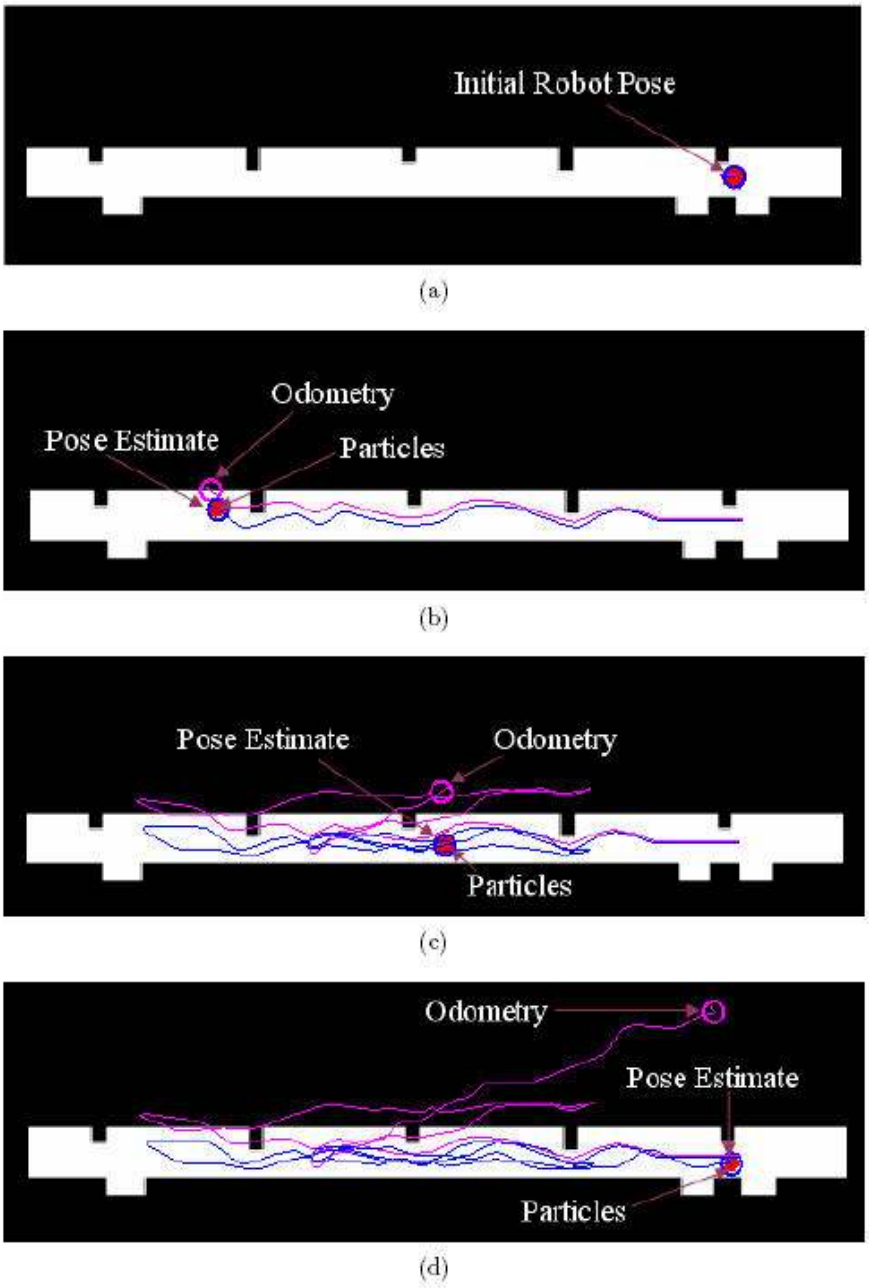


Fig. 4. Implementation of the particle filter to solve the localization problem. Notice that the error from the odometry grows as the robot travels a greater distance.

CONCLUSION

A mobile robot has to possess three competencies to achieve full autonomy: navigation, map building and localization. Over the years, many algorithms have been proposed and implemented with notable success to give mobile robots all the three competencies. Some of the key algorithms such as the *navigation function*, *roadmaps*,

artificial potential field, *vector field histogram*, *hybrid navigation* and the *integrated algorithm* for navigation; *occupancy grid* and *topological* based mapping; as well as the *Kalman filter* and *particle filter* for localization are reviewed in both Part I and II of this article.

FUTURE TRENDS

While the navigation, map building and localization algorithms are implemented with notable success, the scale and structure of the environments for these algorithms to work are limited. Hence, the future challenges for mobile robot autonomy are in the implementations of the algorithms in larger scale and more complex environments such as the urban cities or jungles.

REFERENCES

A. Elfes. (1987). Sonar-based real-world mapping and navigation. IEEE Journal of Robotics and Automation. RA-3(3):249–265.

A. Elfes. (1989). Occupancy grids: A probabilistic framework for robot perception and navigation. PhD thesis, Department of Electrical and Computer Engineering, Carnegie Mellon University.

A.Kelly. (1994). A 3d state space formulation of a navigation Kalman filter for autonomous vehicle. Technical Report. The Robotics Institute, Carnegie Mellon University, 5000 Forbes Avenue, Pittsburgh, PA 15213.

C. Kwok, D. Fox, & M. Meila. (2002). Real-time particle filters. Advances in Neural Information Processing Systems.

D. Fox, C. Kwok, & M. Meila. (2003). Adaptive real-time particle filters. Proceedings of the International Joint Conference on Robotics and Automation (ICRA).

D. Kortenkamp, & T.Weymouth. (1994). Topological mapping for mobile robots using a combination of sonar and vision sensing. Proceedings of the Twelfth National Conference on Artificial Intelligence. pp. 979–984. AAAI Press/MIT Press.

G. Welch, & G.Bishop (1995). An introduction to the Kalman filter. Paper TR 95-041. Department of Computer Science, University of North Carolina at Chapel Hill, Chapel Hill, NC 27599-3175.

H. Choset. (1996). Sensor based motion planning: The hierarchical generalized voronoi graph. PhD thesis, California Institute of Technology.

H. Choset, & J.W. Burdick. (1996). Sensor based planning: The hierarchical generalized voronoi graph. Proceeding Workshop on Algorithmic Foundations of Robotics.

H.P. Moravec. (1988). Sensor fusion in certainty grids for mobile robots. AI Magazine. pp. 61-74.

H.P. Moravec, & D.W. Cho. (1989). A Bayesian method for certainty grids. AAAI 1989 Spring Symposium on Mobile Robots.

Ioannis M. Rekleitis. (2004). A particle filter tutorial for mobile robot localization. Technical Report TR-CIM-04-02, Centre for Intelligent Machines, McGill University, 3480 University St., Montreal, Quebec, CANADA H3A 2A7.

J. Borenstein, & L. Feng. (1995). Umbmark: A benchmark test for measuring odometry errors in mobile robots. Proceedings of the 1995 SPIE Conference on Mobile Robots. pp. 569-574.

J. Borenstein, & L. Feng. (1996). Measurement and correction of systematic odometry errors in mobile robots. IEEE Transaction on Robotics and Automation. Vol. 12, pp. 869-880.

Jodo Xavier, Daniel Castrot, Marco Pachecot, Ant Nio Ruanot, & Urbano Nunes. (2005) Fast line, arc/circle and leg detection from laser scan data in a player driver. Proceedings of the 2005 IEEE International Conference on Robotics and Automation. pp. 95-106.

John J. Leonard, & Hugh F. Durrant-Whyte. (1991). Mobile robot localization by tracking geometric beacons. IEEE Transaction on Robotics and Automation. Vol. 7, pp. 376-382.

R. Chatila, & J.-P. Laumond. (1985). Position referencing and consistent world modeling for mobile robots. Proceedings of the 1985 IEEE International Conference on Robotics and Automation.

R. Smith, M. Self, & P. Cheeseman. (1990). Estimating uncertain spatial relationships in robotics. Autonomous Robot Vehicles. Springer-Verlag. pp. 167–193.

Martin David Adams. (1999). Sensor Modeling, Design and Data Processing for Autonomous Navigation. World Scientific. Vol. 13.

Sen Zhang, Lihua Xie, Martin Adams, & Fan Tang. (2003). Geometrical feature extraction using 2d range scanner. The Fourth International Conference on Control and Automation.

S. Thrun, D. Fox, W. Burgard, & F. Dellaert. (2001). Robust monte carlo localization for mobile robots. Artificial Intelligence Journal (AIJ).

S. Thrun, Wolfram Burgard, & Dieter Fox. (2005). Probabilistic robotics. Cambridge Massachusetts, London, England. The MIT Press.

TERMS AND DEFINTIONS

Odometry: A method to do position estimation for a wheeled vehicle during navigation by counting the number of revolutions taken by the wheels that are in contact with the ground.

Recursive algorithm: It refers to a type of computer function that is applied within its own definition. The *extended Kalman filter* and *particle filter* are recursive algorithms because the outputs from the filters at the current time step are used as inputs in the next time step.

Gaussian distribution: It is also known as normal distribution. It is a family of continuous probability distributions where each member of the family is described by two parameters: mean and variance. This form of distribution is used by the localization with *extended Kalman filter* algorithm to describe the posterior belief distribution of the robot pose.

Jacobians: The Jacobian is a first-order partial derivatives of a function. Its importance lies in the fact that it represents the best linear approximation to a differentiable function near a given point.

Posterior belief: It refers to the probability distribution of the robot pose estimate conditioned upon information such as control and sensor measurement data. The *extended Kalman filter* and *particle filter* are two different methods for computing the posterior belief.

Predicted belief: It is also known as the prior belief. It refers to the probability distribution of the robot pose estimate interpreted from the known control data and in the absence of the sensor measurement data.

Curse of dimensionality: This term was first used by Richard Bellman. It refers to the problem of exponential increase in volume associated with adding extra dimensions to a mathematical space.