Particle Filter Based Self-Localization Using Visual Landmarks and Image Database

Wardah Inam
Hamilton Institute
National University of Ireland
Maynooth, Kildare, Ireland

Abstract— **This paper presents an approach to vision-based self-localization using the combination of particle filter and preprocessed image database. The robot uses particle filter with odometery data and landmark pose for position tracking. Furthermore, it uses the image database to globally localize itself when it has been kidnapped or when no landmark is visible.**

I. INTRODUCTION

This paper describes an approach to self-localization using particle filter and image database. In many applications, it is very important for the robot to know its pose i.e. its position and orientation with respect to its surroundings [1]. Localization can be divided into two general categories; position tracking [5] and global localization [2]. Both are dealt within this paper. A new approach to global localization especially the kidnap robot problem [4] is developed using Hough transformed cross-correlated database of images. Earlier attempts have been made to solve this problem e.g. by sensor resetting [8] and introduction of new uniformly distributed samples [5]. Our proposed method not only helps solve the kidnap robot problem but also helps in localization without an initial position and when no landmarks can be seen in the image.

Particle filters are Monte Carlo Localization Algorithms (MCL) [9]. Particle filters use a Bayesian approach to update the probability distribution function according to the locomotion and sensor readings. Bayes rule is used to estimate the posterior distribution where the input is a sequence of motion and observation. The location of the robot is modelled as the density of set of particles. Each particle consists of robot pose with an associated weight depending on the quality of the prediction of the robot pose $s = [\boldsymbol{x}^i; w^i]$, where $i = 1, 2, ..., M$, s is the set of particles, $\boldsymbol{x}$ is the state of the robot given by $[x, y, \theta]$, w is the weight and M is the number of particles used to represent the possible robot positions.

In this paper particle filter is used to determine the position of 'NAO' robot on a soccer field.These robots are used in the standard platform league of RoboCup. The soccer field is $6050mm \times 4050mm$. There are two goal posts at $0mm$ and $6050mm$ which are yellow and blue to distinguish them [7]. The goal posts are being used as landmarks to determine the pose of the robot. The Field of View of the camera is only 45° so there is less chance of observing more than one landmark at a given time. As more and more landmarks are seen the estimate becomes more accurate. The position of the robot is found with respect to one goal post which gives an estimate of the arc on which the robot could be present Fig. 7. The motion model with relative coordinates of the robot are used to estimate the position of the robot. Also, we have used two goal posts to estimate the position of the robot which uses the motion model with absolute coordinates.

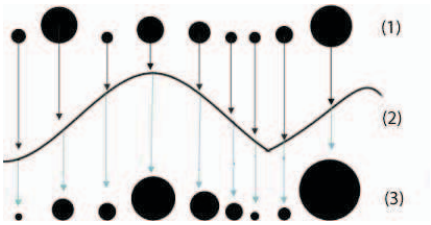


Fig. 1. Particle propagation. (1) Particles at previous state. (2) Pdf of current state. (3) Particles at current state.

A. Methodology

At the beginning, the robot is unaware of its initial position. It uses the Hough transformed cross-correlated image database to find its initial position. An alternative approach used is to spread the particles uniformly all across the field and reiterate until the particles converge. However, for this technique more particles have to be introduced [11]. As the robot moves it uses the particle filter to localize itself with the help of the goal posts. When the goal posts cannot be seen or the robot is kidnapped the robot uses the database to relocate itself. Thus, using the combination of the particle filter and Hough Transformed cross-correlated image database it localizes itself.

B. Background

We model the robot motion as a Markovian process where the likelihood of future state depends on the present state and not past states. The particles are moved according to the motion model. Depending on the likelihood of the pose given by the posterior distribution function the weights of the particles change which is shown in Fig. 1 by changing size. The posterior distribution is given by

$$p(x_t|D_t) = \frac{p(z_t|x_t)p(x_t|D_{t-1})}{p(z_t|D_{t-1})} \quad (1)$$

where

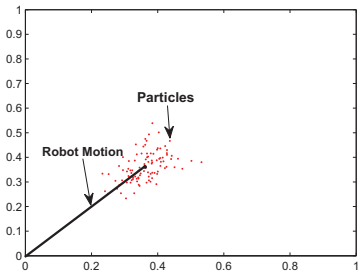


Fig. 2. Prediction

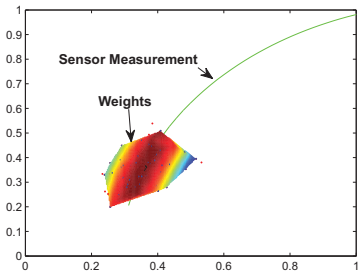


Fig. 3. Update

x_t: Robot state at time t.
z_t: Sensor reading at time t.
u_t: Odometery data at time t.
D_t: z_t and u_t combined at time t.
This pdf can be calculated recursively by finding

$$p(x_t|D_{t-1}) = \int p(x_t|x_{t-1},u_{t-1})p(x_{t-1}|D_{t-1})dx_{t-1} \quad (2)$$

where $p(x_t|x_{t-1},u_{t-1})$ is obtained from the motion model and $p(x_{t-1}|D_{t-1})$ is the pdf from the last step. The normalizing denominator is

$$p(z_t|D_{t-1}) = \int p(z_t|x_t)p(x_t|D_{t-1})dx_t \quad (3)$$

where $p(z_t|x_t)$ is obtained from the sensor model

The particle filter algorithm consists of three general steps. The first step is *prediction* which involves acquiring odometry data from the sensors of the robot and projecting the particles according to the odometery data and motion model. The second step is *update* where we acquire sensor data and normalize the weights of the particles accordingly. The final step is *resampling* in which we duplicate and reject the particles according to their weights when need arises. The details of these steps will be discussed is the following sections.

II. Prediction

A. Action Model

The action model consists of a motion model and a noise model. The motion model depends on robot kinematics and the odometery data. NAO is a bipedal robot capable of

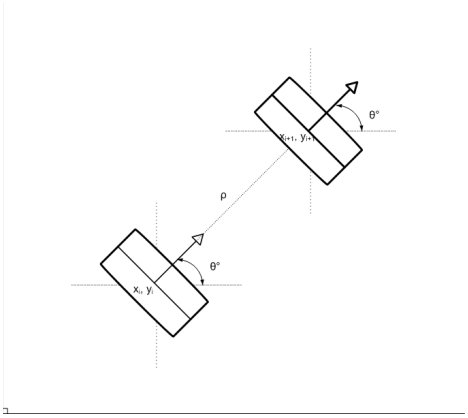


Fig. 4. Motion of Robot

forward translation, rotation and sideways steps. The three motions have been included in the motion model. At a particular instant of time the state of the robot is given by $\mathbf{x} = [x; y; \theta]$ where θ is its orientation as seen in Fig. refmotion. When a robot moves its state changes depending on the motion model.It moves a distance $d\rho$ with a change in angle $d\theta$. The motion model is given by

$$\begin{bmatrix} \theta_{i+1} \\ x_{i+1} \\ y_{i+1} \end{bmatrix} = \begin{bmatrix} \theta_i + d\theta_i \\ x_i + d\rho_i \cos(\theta_{i+1}) \\ y_i + d\rho_i \sin(\theta_{i+1}) \end{bmatrix} \quad (4)$$

Also, noise has to be accounted for in the odometery data. This includes slippage, drift while translation etc [6]. The noise model used is Gaussian with a zero mean and standard deviation relative to the amount of rotation and translation. The action model is given by

$$\begin{bmatrix} \theta_{i+1} \\ x_{i+1} \\ y_{i+1} \end{bmatrix} = \begin{bmatrix} \theta_i + d\theta_i + \varepsilon_{\theta i} \\ x_i + (d\rho_i + \varepsilon_{transi}) \cos(\theta_{i+1} + \varepsilon_{drift_i}) \\ y_i + (d\rho_i + \varepsilon_{transi}) \sin(\theta_{i+1} + \varepsilon_{drift_i}) \end{bmatrix} \quad (5)$$

where $\varepsilon_\theta = \aleph(\mu_\theta, \sigma_\theta * d\theta)$, $\varepsilon_{trans} = \aleph(\mu_{trans}, \sigma_{trans} * d\rho)$ and $\varepsilon_{drift} = \aleph(\mu_{drift}, \sigma_{drift} * d\rho)$. Where ε_i is independent of ε_j for $j \neq i$.

III. Update

A. Observation Model

The vision system on the robot can predict the relative pose of the robot with respect to visual landmarks. Sensor noise has been added in the observation model. The observation model considers two main cases i.e. observing one goal post and observing two goal posts of the particular goal. Thus, the robot can determine its relative pose from the four goal posts on the field.

1) One Landmark: For one goal post relative coordinates of the robot to the goal post are considered. The height of the post gives the distance of the robot from the goal post denoted by ρ. The distance of the post from the center of the image is given by d which is shown in Fig. 5 this gives us the relative angle of the robot to the goal post. A

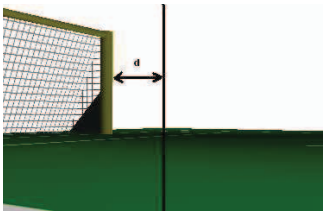


Fig. 5. Relative angle from one goal post

single position of the robot cannot be obtained from this observation. However, the probable positions of the robot from the motion model along with the estimate of the arc on which the robot can be present are used to assign the weights as seen in Fig. 7.

2) Two Landmarks: Similar to one landmark case the height of the goal post gives ρ. ρ_1 and ρ_2 as seen in Fig. 6 are the distances from the specific goal post being observed. ϕ is found by the cosine rule using the values of ρ_1, ρ_2 and the distance between the goal post. θ_e is the estimate of the orientation of the robot found from the image.

$$\begin{bmatrix} x \\ y \\ \theta \end{bmatrix} = f(\rho_1, \theta_e, \phi) \tag{6}$$

Using the observation model and sensor data the weights of the particles can be assigned.

B. Finding Weight

The weight of each particle is proportional to its probability. The action model and sensor model effect the weight of the particles. The probable position of the particle is found for one goal post case by

$$W \propto \frac{1}{\sqrt{2\pi}\sigma_\rho} e^{\frac{-(\rho-\rho')^2}{2\sigma_\rho{}^2}} \frac{1}{\sqrt{2\pi}\sigma_\phi} e^{\frac{-(\phi-\phi')^2}{2\sigma_\phi{}^2}} \tag{7}$$

However, for a two goal post case we use a multivariate Gaussian as ϕ is dependent on ρ_1 and ρ_2

$$W \propto \frac{1}{(2\pi)^{\frac{3}{2}}|\Sigma|^{\frac{1}{2}}} e^{\frac{-1}{2}(\mathbf{x}-\mathbf{x}')\Sigma^{-1}(\mathbf{x}-\mathbf{x}')^T} \tag{8}$$

where Σ is the covariance matrix.

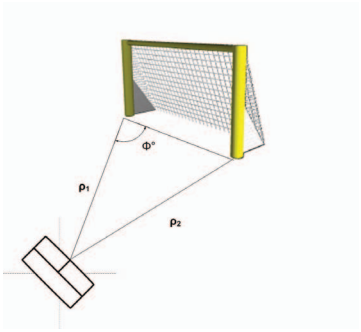
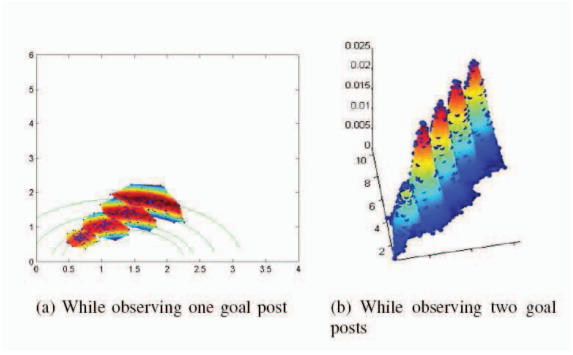


Fig. 6. Using two goal posts for self-localization



(a) While observing one goal post (b) While observing two goal posts

Fig. 7. Forward motion of the robot with the colour spectrum and height of particles representing the weight at each step.

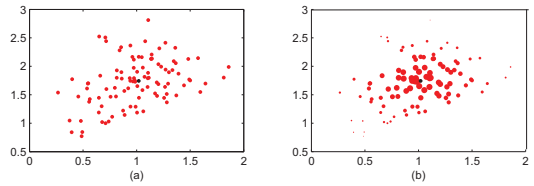


Fig. 8. Particle size before (a) and after (b) resizing according to weight. The 'black dot' is the weighted mean of the particles which gives the pose of the robot

C. Normalization of Weights

As the normalization constant is unknown (3) the weights have to be normalized [6]. This is done by summing the weights and dividing each weight by the sum.

D. Weighted Mean

To get the pose of the robot the weighted mean is calculated as

$$x_m = \sum_{i=1}^{M} x_i \cdot w_i \tag{9}$$

In Fig. 8 the particle size is proportional to its weight and the weighted mean gives its pose.

IV. Resampling

The resampling is done by generating M uniformly distributed random numbers. These numbers are sorted and the cumulative sum of the weights of the particles is found as shown in Fig. 9. The sorted numbers are projected to the sum and the particles corresponding to the weights are selected. The particles are copied equal to the number of projections and other particles are deleted. Thus, the number of particles remains constant after resampling.

Without resampling, when the particles are projected forward the cluster of particles spreads as shown in Fig. 10. The colour spectrum which has been linearly interpolated shows the weights of the particles. If resampling is not done only a single particle with very high weight will be left with the rest of the particles having very low weight. This is not a good representation of a pdf. Thus, the particles

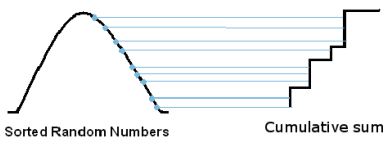


Fig. 9. Resampling

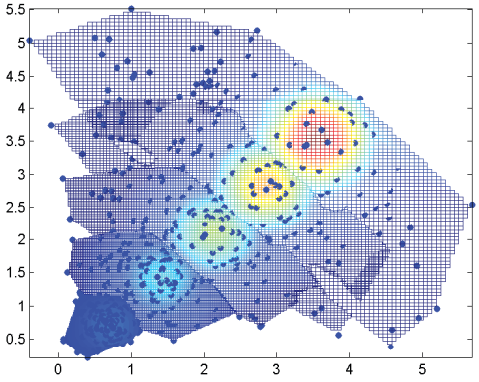


Fig. 10. Robot motion without resampling

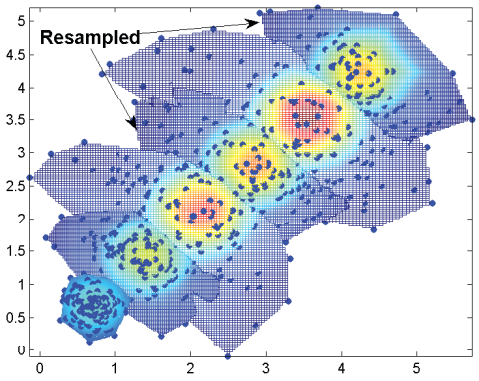


Fig. 11. Robot motion with Resampling

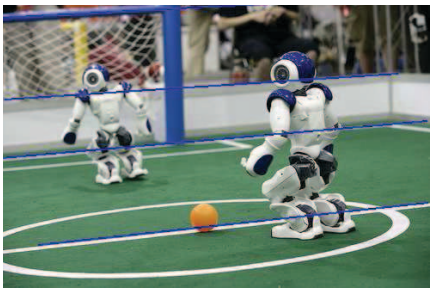


Fig. 12. Hough lines example

have to be resampled continuously. Resampling can also be done at each step, however, this will slow down the process. So, the effective sample size E [6] is found.If the effective sample size falls below some predetermined threshold then the particles are resampled. The effective sample size has an inverse relationship with the coefficient of variation C [6].

$$E = \frac{M}{1+C} \tag{10}$$

$$C = \frac{1}{M}\sum_{i=1}^{M}(Mw(i)-1)^2 \tag{11}$$

V. Hough Transformed Cross-Correlated Image Database

The first and foremost problem associated with relying on computer vision for localization is the presence of noise and obstructions in image data. The Hough Transform [10], a well known computer vision tool used to detect straight lines and other curves in images, is used to aid in overcoming this problem. A given image of the pitch may contain a number of interfering objects, such as other robots and the ball. Executing a Hough Transform on the image returns only the straight lines contained within it for example the boundary pitch lines, as shown in Fig. 12.

In previous works these Hough lines, as well as other landmarks, are used to aid in calculating robots location [11]. The fundamental idea in this paper is to match the current viewpoint of the robot against a database of all possible viewpoints to give an absolute position. The database contains no interfering objects, only viewpoints of an empty field. Use of the Hough Transform on the robot's in-game viewpoint virtually removes any interfering objects, allowing matching of pitch boundary lines against a database of pitch boundary lines.

To implement this scheme involves two stages. Preprocessing is carried out before a game begins, and online processing takes place during the game. During the preprocessing the pitch is divided into a grid of mxn possible positions as seen in Fig. 13(a). A 360° panorama of each grid position, using n° increments (Equivalent to $360/n$ orientations) is taken as seen in Fig. 13(b). Each panorama is processed to leave a database of Hough Transform lines as seen in Fig. 13(c).

The density of the grid and panoramas in the database affect both performance and memory load. A higher density grid provides a more accurate position, but increases match time by a significant factor. A higher density panorama image increases accuracy in determining orientation, but again increases time taken to find a match. For a pitch divided into mxn positions, with n° incremented panoramas, the location can be approximated to one of mxn possibilities with an orientation approximation of n possibilities for each of the mxn locations. Taking a 4x6 grid with 45° incremented panoramas as an example, a final full processed database will

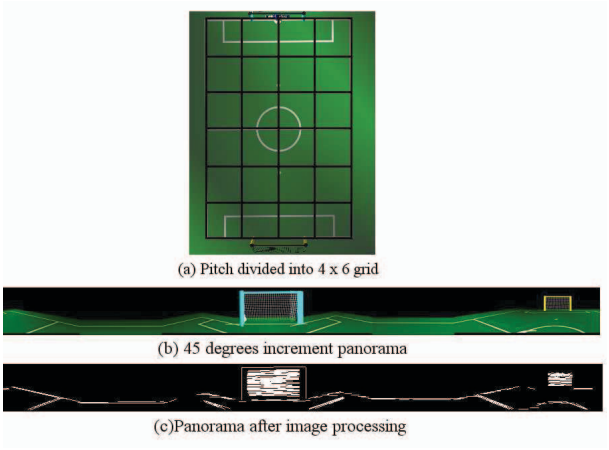


Fig. 13.

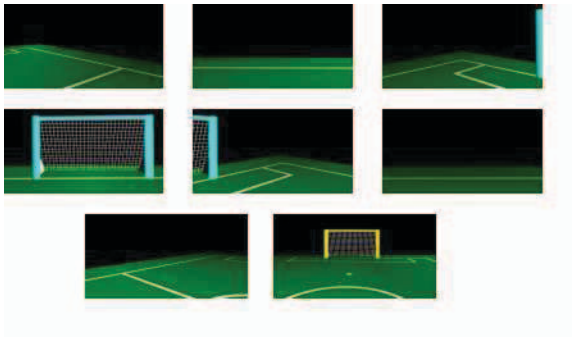


Fig. 14. Eight viewpoint orientations matched

Fig. 15. Only the end points of the Hough Lines

contain 24 potential locations, with 8 potential orientations each, a total of 192 positions. This database can then be stored as a collection of 24 images onboard the robot.

During the game the processing takes place on the robot. The image is received from one of the cameras of the robot. This is processed to leave only Hough Transform lines. These lines are cross-correlated against all potential positions and orientations in the database. Location of the maximum correlation is determined to be the position and orientation of the robot.

To produce a Hough transformed image the Blue Channel is extracted from the RGB image which aids in differentiating between the blue and yellow goals as well as removing the orange ball from the image. An intensity threshold is performed to highlight the pitch lines and discard other features not being used. Edge detection is carried out which is followed by the Hough Transform.

VI. Results

A testing environment was developed for this technique with Intel's OpenCV through C++. The test system was an Intel Pentium D 3.0Ghz Dual Core, with 1GB of RAM, running Microsoft Windows XP SP3. The test data was a scaled down version of the full database, with simulated signal data. The images were acquired from Cyberbotics Webots simulator for RoboCup. A single grid position was simulated, with 45° orientation increments. There are two ways of using the output of the Hough Transform. One way involves using the full lines returned by the function as shown in Fig. 13, and the other involves using only the end points of these lines, shown in Fig. 15. Depending on whether the lines or endpoints are used, the database and in-game viewpoints must be processed the same way. Using a database of 901x71 pixels and Viewpoints of 121x66 pixels with full Hough Transform lines, the match-up time was 0.0169s while the accuracy was 7/8. While using endpoints of Hough Transform lines the match-up time remained the same however the accuracy was 100%.

Based on these results it is clear that this Hough Transformed database approach is a viable solution for global robot localization, in this setting. Though there are some problems associated with this method such as large obstructions to the field of view will hinder this method as some line segments need to be visible to attempt correlation. Also, using a low density grid or panorama can cause approximation problems. Given the large amount of cross-correlation occurring a substantial amount of processing time is involved. A higher density database is more accurate, but can heavily impact performance, as well as memory requirements. Specifically in regards to implementation in NAO robots that are limited to a 500MHz processor, and 256MB of RAM. Storing image information as standard image data for example with a 4x6, 45° incremented database, containing 160x120 sized viewpoints, using a 1-channel 8-bit grayscale format is of size 29.5MB. However, using a specialist binary format akin to a two dimensional array containing single bit values would result in a database of size 3.6MB. Compression techniques can be used to further reduce the file size, however, on-board decompression will slow down the process.

VII. CONCLUSION

In this paper a new approach for vision based localization using the combination of particle filter and image database has been presented. Our method switches from using the odometery data with the motion model in the particle filter to independently finding the pose of the robot using the database when the need to globally estimate the pose of the robot arises. This approach would be highly useful for robots in RoboCup where we know the environment and have time prior to the competition to develop the database. With this system, the robot is not only able to locally track its position with the help of visible landmarks but also globally localize itself by using an image database.

VIII. Acknowledgments

The author would like to acknowledge Thomas Whelan for his contribution to the paper especially for his work on the Hough transformed image data base; Richard Middleton and Mark Verwoerd for their guidance and support; Dr. M. Junaid Mughal and Dr. Nisar Ahmed for their encouragement; Ghulam Ishaq Khan Institute of Engineering Sciences and Technology, Pakistan; National University of Ireland, Maynooth; Science Foundation of Ireland and the UREKA program for giving the opportunity and the facilities.

References

[1] F. Dellaert, D. Fox, W. Burgard, and S. Thrun, "Monte carlo localization for mobile robots". In Proceedings of the IEEE International Conference on Robotics and Automation (ICRA), 1999.

[2] Guanghui Cen; Matsuhira, N.; Hirokawa, J.; Ogawa, H.; Hagiwara, I., "Mobile robot global localization using particle filters," Control, Automation and Systems, 2008. ICCAS 2008. International Conference on , vol., no., pp.710-713, 14-17 Oct. 2008.

[3] Gordon, N.J.; Salmond, D.J.; Smith, A.F.M., "Novel approach to nonlinear/non-Gaussian Bayesian state estimation," Radar and Signal Processing, IEE Proceedings F, vol.140, no.2, pp.107-113, Apr 1993

[4] S. Thrun, D. Fox, W. Burgard, and F. Dellaert, Robust monte carlo localization for mobile robots, Artificial Intelligence Journal, 2001.

[5] D. Fox, W. Burgard, F. Dellaert, and S. Thrun, Monte carlo localization: Efficient position estimation for mobile robots, in Proc. of the National Conference on Artificial Intelligence, 1999.

[6] Ioannis Rekleitis. A Particle Filter Tutorial for Mobile Robot Localization. Technical Report TR-CIM-04-02, Centre for Intelligent Machines, McGill University, Montreal, Quebec, Canada, 2004.

[7] RoboCup Standard Platform League (Nao) Rule Book RoboCup Technical Committee.

[8] Lenser, S.; Veloso, M., "Sensor resetting localization for poorly modelled mobile robots," Robotics and Automation, 2000. Proceedings. ICRA '00. IEEE International Conference, vol.2, no., pp.1225-1232 vol.2, 2000.

[9] D. Fox, S. Thrun, W. Burgard, and F. Dellaert. Particle filters for mobile robot localization. In A. Doucet, N. de Freitas, and N. Gordon, editors, Sequential Monte Carlo Methods in Practice. Springer, 2001.

[10] Richard O. Duda and Peter E. Hart, "Use of the Hough Transformation To Detect Lines and Curves in Pictures", 1972.

[11] Hauke Strasdat, Maren Bennewitz, and Sven Behnke,"Multi-Cue Localization for Soccer Playing Humanoid Robots" , 2006.

[12] Maskell, S.; Gordon, N., "A tutorial on particle filters for on-line nonlinear/non-Gaussian Bayesian tracking," Target Tracking: Algorithms and Applications (Ref. No. 2001/174), IEE , vol.Workshop, no., pp. 2/1-2/15 vol.2, 16-17 Oct. 2001.

[13] Carpenter, J.; Clifford, P.; Fearnhead, P., "Improved particle filter for nonlinear problems," Radar, Sonar and Navigation, IEE Proceedings, vol.146, no.1, pp.2-7, Feb 1999.

[14] Wolf, J.; Burgard, W.; Burkhardt, H., "Robust vision-based localization by combining an image-retrieval system with Monte Carlo localization," Robotics, IEEE Transactions on , vol.21, no.2, pp. 208-216, April 2005.

[15] Guanghui Cen; Nakamoto, H.; Matsuhira, N.; Hagiwara, I., "Effective application of Monte Carlo localization for service robot," Intelligent Robots and Systems, 2007. IROS 2007. IEEE/RSJ International Conference on , vol., no., pp.1914-1919, Oct. 29 2007-Nov. 2 2007.

[16] J.Borenstein, B.Everett, and L.Feng. Navigating Mobile Robots: Systems and Techniques, A.K. Peters Ltd., Wellesley, MA, 1996.