Probabilistic Cooperative Localization and Mapping in Practice

Ioannis Rekleitis[1], Gregory Dudek[1] and Evangelos Milios[2]

[1]Centre for Intelligent Machines, McGill University, Montreal, Québec, Canada.
[2]Faculty of Computer Science, Dalhousie University, Halifax, Nova Scotia, Canada
contact: {yiannis,dudek}@cim.mcgill.ca, eem@cs.dal.ca

Abstract

In this paper we present a probabilistic framework for the reduction in the uncertainty of a moving robot pose during exploration by using a second robot to assist. A Monte Carlo Simulation technique (specifically, a Particle Filter) is employed in order to model and reduce the accumulated odometric error. Furthermore, we study the requirements to obtain an accurate yet timely pose estimate. A team of two robots is employed to explore an indoor environment in this paper, although several aspects of the approach have been extended to larger groups. The concept behind our exploration strategy has been presented previously and is based on having one robot carry a sensor that acts as a "robot tracker" to estimate the position of the other robot. By suitable use of the tracker as an appropriate motion-control mechanism we can sweep areas of free space between the stationary and the moving robot and generate an accurate graph-based description of the environment. This graph is used to guide the exploration process. Complete exploration without any overlaps is guaranteed as a result of the guidance provided by the dual graph of the spatial decomposition (triangulation) of the environment. We present experimental results from indoor experiments in our laboratory and from more complex simulated experiments.

1 Introduction

In this paper we consider the application of probabilistic methods to the problem of exploration by a pair of robots (i.e. collaborative exploration [11]). In particular, we consider the particular parameters that allow us to efficiently carry out collaborative exploration using a particle system to model the robots pose and uncertainties. In the last several years, particle filtering (also known as *condensation* [6]) has been demonstrated to be an effective method for non-parametrically estimating the parameters and uncertainty of systems of moderate complexity, in particular but not exclusively when the uncertainty of the system exhibits a multi-modal probability distribution. While particle systems for modeling uncertainty have clear theoretical advantages, they can be substantially slower than parametric methods such as Kalman filters in practice. Further, while they are touted as having better tracking and convergence properties than Kalman Filters (for example) – a certainty in principle – in reality with a finite number of particles this depends critically on suitable parameters that trade off this idealized robustness for some measure of efficiency. These trade-offs relate to the number of particles used, the resampling strategy, and several related parameters.

Figure 1: The two robots exploring one side of an office building. On the left a robot carries the three plane range target, on the right a robot with a laser mapping a reflex corner (using sonar).

In this paper we will present a case study that illustrates the particular trade-offs necessary to achieve both acceptable speed and good accuracy in the context of collaborative exploration. We also consider a simple yet non-standard model of odometry error that appears more appropriate than those usually employed.

This work builds on our prior results in which we define the problem of collaborative exploration in which a team of two or more robots coordinate their motion through a potentially unknown environment to jointly estimate one another's position and, in so doing, estimate the layout of the environment of any spatial parameter of interest. The key to collaborative exploration is to have at least one "tracker" sensor that allows a robot to estimate the positions of other robots in the team. This allows inter-robot sensing to compensate for arbitrarily large odometry errors, as well as presenting other advantages [11, 13]. Our specific strategy for collaborative

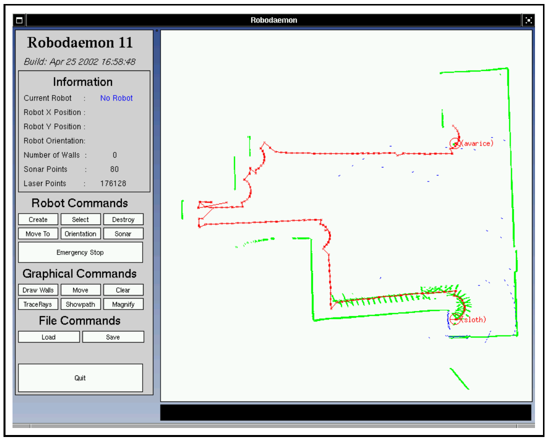


Figure 2: The trajectories of the two robots with the laser data also marked. Note the target pattern detected.

exploration as applied to a team of two robots is to have the robots take turns moving so that any any time one is stationary and can act as a fixed reference point. While doing this we estimate the positions of the robots using a particle filter that combines an open-loop estimate of odometry error with sensor data collected from the tracker, a LIDAR-based laser range finder on one robot and a three plane target mounted on top of the second robot (alternative implementations have been used in prior work). Figure 1a show the two robots during the exploration of a simple environment; the three plane target is visible from any orientation as can be seen in Figure 2 where the sensor data from the laser range finder are recorded over time. Parts of the walls are also mapped by the laser and provided ground truth for the cooperative exploration.

2 Particle Filtering

Different methods have been employed in the past in order to estimate and reduce the uncertainty of a moving robot [8, 14]. One approach that has gained popularity of late falls under the category of Monte Carlo Simulation (see [4] for an overview) and is known under different names in different fields. The technique we use was introduced as particle filtering by Gordon *et al.* [5]; in mobile robotics, particle filtering has been applied successfully by different groups [3, 7, 15]. In vision this technique was introduced under the name of condensation [6] and particle filtering [1]. The general outline of this approach is described bellow.

The main objective of particle filtering is to "track" a variable of interest as it evolves over time. A series of actions are taken, each one modifying the state of the variable of interest according to some model. Moreover, at certain times an observation arrives that describes the state of the variable of interest at that time.

Multiple copies (particles) of the variable of interest are used, each one associated with a weight that signifies the quality of that specific particle. A description of the variable of interest is obtained by the weighted sum of all the particles. After each action, each particle is modified according to the existing model (prediction stage), including the addition of random noise in order to simulate the effect of noise on the variable of interest. Then, each particles weight is reevaluated based on the latest sensory information available (update stage). At times the particles with infinitesimally small weights are eliminated, a process called resampling.

More formally, the variable of interest (in our case the pose of the moving robot $\mathbf{x}^k = [x^k, y^k, \theta^k]^T$) at time $t = k$ is represented as a set of N samples (usually called "particles") ($S_i^k = [\mathbf{x}_j^k, w_j^k] : j = 1 \ldots N$) [1] each one consisting of a copy of the variable of interest and a weight (w_j^k) that defines the contribution of this particle to the overall estimate of the variable. The particle filter algorithm is recursive in nature and operates in two phases: *prediction* and *update*. Algorithm 1 presents a formal description of the particle filter algorithm and the next two subsections discuss the details of prediction and update.

Require: A set of Particles for Robot i at time 0: $S_i^0 = [\mathbf{x}_j, w_j : j = 1 \ldots N]$.
$W = w_j : j = 1 \ldots N$
while (Exploring) **do**
  $k = k + 1$;
  if (<u>ESS</u>(W) $< \beta * N$) **then** {Particle Population Depleted (Equation 7)}
    Index=<u>Resample</u>(W);
    $S_i^k = S_i^k(Index)$;
  end if
  for ($j = 1$ to N) **do** {Prediction after action α}
    $\mathbf{x}_j^{k+1} = \hat{f}(\mathbf{x}_j^k, \alpha)$
  end for
  s=<u>Sense</u>()
  for ($j = 1$ to N) **do** {Update the weights}
    $w_j^{k+1} = w_j^k * \mathcal{W}(s, \mathbf{x}_j^{k+1})$
  end for
  for ($j = 1$ to N) **do** {Normalize the weights}
    $w_j^{k+1} = \frac{w_j^{k+1}}{\sum_{j=1}^{N} w_j^{k+1}}$
  end for
end while

Algorithm 1: Particle Filter Algorithm; functions are noted as underlined text, Comments are inside curly brackets.

2.1 Prediction

If at time $t = k$ we know the probability distribution function (*pdf*) of the system with respect to position at the previous instant (time $t = k - 1$) then we model the effect

[1] The index j denotes the particle and not the robot.

of the action α to obtain a prior estimate of the *pdf* at time $t = k$ (prediction). In other words, the *prediction* phase uses a model in order to simulate the effect an action has on the set of particles with the appropriate noise added as in equation 1.

$$S^k = f(S^{k-1}, \alpha, \mathbf{v}) \tag{1}$$

where $\mathbf{v}$ is the added noise.

In our case the variable of interest is the pose of the moving robot and each action α is a motion by $(\Delta x, \Delta y)$; such a motion could be performed as a rotation followed by a translation. If the robot's initial pose is $[x, y, \theta]^T$, then the robot first rotates by $\delta\theta = \theta_k - \theta$, where $\theta_k = \arctan(\Delta y, \Delta x)$ to face the destination position, and then it translates forward by distance $\rho = \sqrt{\Delta x^2 + \Delta y^2}$ [2]. If the starting pose is $[x, y, \theta]^T$, the resulting pose $[x', y', \theta_k]^T$ is given in equation 2. Consequently, the noise model is applied separately in each of the two types of motion since they are independent.

$$\begin{bmatrix} x' \\ y' \\ \theta' \end{bmatrix} = \begin{bmatrix} x + \rho\cos(\theta_k) \\ y + \rho\sin(\theta_k) \\ \theta_k \end{bmatrix} \tag{2}$$

When the robot performs a relative rotation by $\delta\theta$ the noise from the odometry error is modeled as a Gaussian with mean M_{rot} experimentally established and sigma proportional to $\delta\theta$. Therefore, to model the rotation of $\delta\theta$ the orientation θ_j of each particles j is updated by adding $\delta\theta$ plus a random number drawn from $N(M_{rot}, \sigma_{rot}\delta\theta)$.

The error model for the translation is more complicated. Two different sources of error are modeled, the first is related to the actual distance traveled and the second is related to changes in orientation during the forward translation. In particular, during the translation the orientation of the robot changes constantly resulting in a deviation from the direction of the translation; such effect is called drift and is modeled by adding a small amount of noise in the orientation of the robot before and after each step. As well, if the intended distance is ρ, the actual distance traveled is given by ρ plus some noise. Experimental results provide the expected value and the standard deviation for the drift and pure translation. Because it is very difficult to analytically model the continuous process, a simulation is used that discretizes the motion to "K" steps. If $[\sigma_{trans}, \sigma_{drift}]$ are the experimentally obtained values per distance traveled then at each step of the simulation the standard deviation is given in equation 3.

$$\sigma_{trs} = \sigma_{trans}\sqrt{K}, \text{ and } \sigma_{drft} = \sigma_{drift}\sqrt{\tfrac{K}{2}} \tag{3}$$

Equation 4 presents the precise model for updating the pose after one step in translation (out of a total of"K" steps).

[2]In our experimental setup the super Scouts robots used are controlled by the same rules.

$$\overline{\mathbf{x}}_{i+1} = \begin{pmatrix} x_{i+1} \\ y_{i+1} \\ \theta_{i+1} \end{pmatrix} = \begin{pmatrix} x_i + (\Delta\rho + \mathcal{E}_{\Delta\rho})\cos(\theta_i + \mathcal{E}_{\theta_1}) \\ y_i + (\Delta\rho + \mathcal{E}_{\Delta\rho})\sin(\theta_i + \mathcal{E}_{\theta_1}) \\ \theta_i + \mathcal{E}_{\theta_1} + \mathcal{E}_{\theta_2} \end{pmatrix}$$

$$\text{where } \begin{pmatrix} \mathcal{E}_{\Delta\rho} \\ \mathcal{E}_{\theta_1} \\ \mathcal{E}_{\theta_2} \end{pmatrix} = \begin{pmatrix} \mathbf{N}(M_{trans}, \sigma_{trans}\sqrt{N}\Delta\rho) \\ \mathbf{N}(\frac{M_{drift}}{2}, \frac{\sigma_{drift}\sqrt{N}\Delta\rho}{\sqrt{2}}) \\ \mathbf{N}(\frac{M_{drift}}{2}, \frac{\sigma_{drift}\sqrt{N}\Delta\rho}{\sqrt{2}}) \end{pmatrix} \tag{4}$$

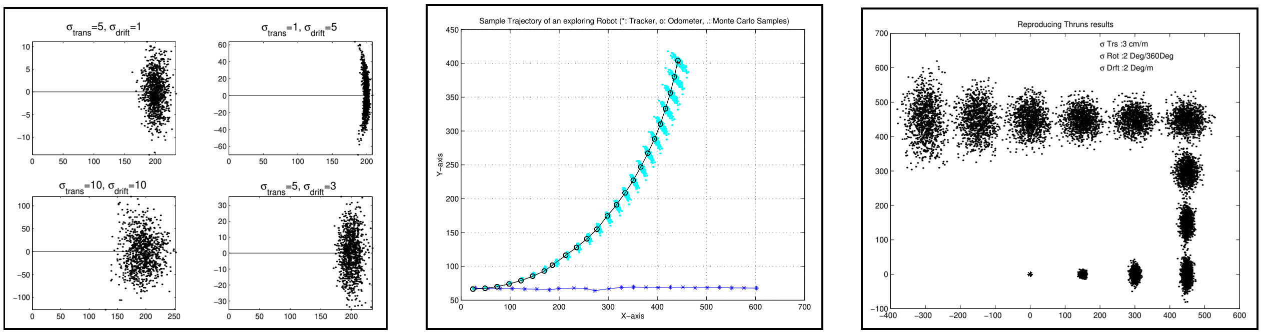
Figure 3a presents a graphical illustration of the effect of the two noise parameters $(\sigma_{trs}, \sigma_{drft})$ in the predictive model. In all cases the robot makes a single forward motion of 200cm. In the top left sub-plot of 3(a), the uncertainty at the distance traveled is much higher than the drift uncertainty and thus the particles spread a lot more in the direction of the motion. In contrast, in the top right sub-plot, where drift noise dominates, the particles spread along an arc. The bottom left sub-plot presents the spread of particles for equal high values of the noise parameters. Finally, the bottom right sub-plot presents the spread of particles for noise parameters collected in our laboratory.

The last two sub-figures of Figure 3 present examples of complex motions and illustrates the performance of the predictive model. Sub-figure 3b presents experimental validation of our predictive model. In this case the predictive model was guided by a set of motion commands that were used in an experiment in our laboratory [3]. In short, the experiment consisted of forward translations, each one followed by four rotations by ninety degrees. The curved trajectory in sub-figure 3b represent the uncorrected odometer values. The odometry estimates deviated due to noise despite the fact that the actual trajectory of the robot was kept in a straight line as indicated by the lower straight line is sub-figure 3b. The predictive model was constructed using the noise statistical parameters collected in our laboratory . The predicted cloud of particles can be seen around the recorded odometry values following the trajectory with high accuracy. In sub-figure 3c, the robot moves forward three times, rotates ninety degrees, then translates forward three more times, after which it rotates again by ninety degrees and translates forward five times. As can be seen the uncertainty grows unbounded.

2.2 Update

The *update* phase uses the information obtained from sensing to update the particle weights in order to accurately describe the moving robot's probability distribution. A measurement from the robot tracker sensor is guaranteed to exist after each motion; for ease of reference we represent this

[3]For the full description of this experiment please refer to [10].



(a) (b) (c)

Figure 3: (a) The effect of different noise parameters for a forward translation of 200cm. (b) Series of forward translations and 360° rotations. (c) Large trajectory, uncertainty build-up.

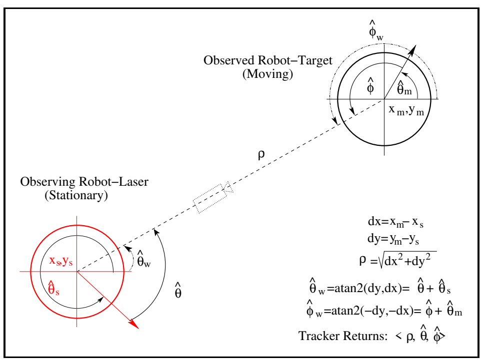


Figure 4: The stationary robot with the robot tracker sensor observes the moving robot that carries the target. Note that the "camera" indicates the robot with the *Robot Tracker*; and $\hat{\theta}_w, \hat{\phi}_w$ are angles in world coordinates.

measurement by the triplet $T = [\rho\ \phi\ \theta]$, where ρ is the distance between the two robots, ϕ is the angle at which the observing robot sees the observed robot relative to the heading of the observing robot, and θ is the heading of the observed robot as measured by the observing robot relative to the heading of the observing robot (see Figure 4). If the stationary robot is equipped with the robot tracker, where $\mathbf{X}_m = [x_m, y_m, \theta_m]^T$ is the pose of the moving robot and $\mathbf{X}_s = [x_s, y_s, \theta_s]^T$ is the pose of the stationary robot then equation 5 returns the sensor output T. Different implementations of the robot tracker sensor are possible. Currently we employ a combination of a laser range finder with a three plane target (see Figure 1a). From any position around the robot with the three plane target, the laser range finder always detects at least two planes, thus being able to recover the pose of the target robot.

$$\begin{bmatrix} \rho \\ \theta \\ \phi \end{bmatrix} = \begin{bmatrix} \sqrt{dx^2 + dy^2} \\ atan2(dy, dx) - \theta_s \\ atan2(-dy, -dx) - \theta_m \end{bmatrix} \tag{5}$$

where $dx = x_m - x_s$ and $dy = y_m - y_s$.

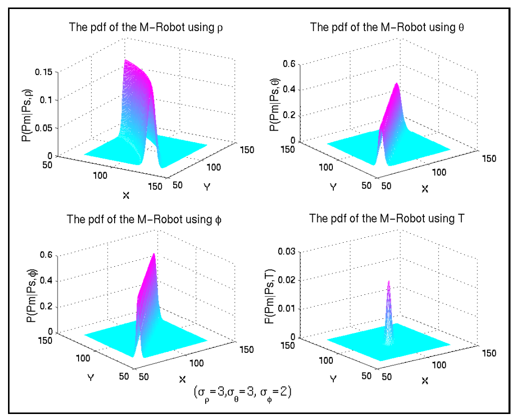


Figure 5: *The contribution of each measurement of the robot tracker in the weighting* pdf *of the moving robot.*

After the sensor measurement ($\mathbf{z}$) becomes available the weights of the particles are updated by $W_i' = P(\mathbf{z}|\mathbf{x}_i)W_i : i = 1 \ldots N$ where the probability of the sensor measurement $\mathbf{z}$ given a particle $\mathbf{x}_i$ is given by equation 6.

$$P(\mathbf{z}|\mathbf{x}_i) = \frac{e^{\frac{-(\rho-\rho_i)^2}{2\sigma_\rho^2}}}{\sqrt{2\pi}\sigma_\rho} \frac{e^{\frac{-(\theta-\theta_i)^2}{2\sigma_\theta^2}}}{\sqrt{2\pi}\sigma_\theta} \frac{e^{\frac{-(\phi-\phi_i)^2}{2\sigma_\phi^2}}}{\sqrt{2\pi}\sigma_\phi} \tag{6}$$

Figure 5 illustrates the conditional probabilities used to compute the pose updates as a function of the disparity between prediction and observation. Equation 6 above presents the composite formula.

2.3 Resampling

One of the problems that appear with particle filters in practice, especially with low particle densities, is the depletion of the particle population in some regions of space after a few iterations. As most of the particles have drifted far enough, their weights become very small and they no longer contribute to estimates of the position of the moving robot. Liu *et al.* [9] propose two measures that estimate the number of *near-zero-weight* particles: the coefficient of variation cv_t^2 and the effective sample size ESS_t (see equation 7).

$$cv_t^2 = \frac{var(w_t(i))}{E^2(w_t(i))} = \frac{1}{N}\sum_{i=1}^{N}(Nw(i)-1)^2,$$
$$\text{and } ESS_t = \frac{N}{1+cv_t^2} \qquad (7)$$

When the effective sample size (ESS) drops below a certain threshold, usually a percentage of the number of particles N, then the particle population is resampled, eliminating (probabilistically) the ones with small weights and duplicating the ones with higher weights. Different methods have been proposed for resampling; we used the approach by Carpenter *et al.* [2] that runs in linear time (see algorithm 2).

```
Input: double W[N]
Require: ∑_{i=1}^{N} W_i = 1
  Q = cumsum(W); {calculate the running totals Q_j = ∑_{l=0}^{j} W_l}
  t = -log(rand(N+1));
  T = cumsum(t); {calculate the running totals T_j = ∑_{l=0}^{j} t_l}
  TN = T/T(N+1);{normalize T to TN;}
  i=1; j=1; {Arrays start at 1}
  while (i ≤ N) do
    if T[i] < Q[j] then
      Index[i]=j;
      i=i+1;
    else
      j=j+1;
    end if
  end while
  Return(Index)
```

Algorithm 2: Linear Time Resampling Algorithm.

3 Experimental Results

The exploration algorithm used for the mapping of an indoor environment is based on the triangulation of free space by the two robots [4]. The line of visual contact is used to "sweep" the space; in other words, if the two robots can observe each other then the space in between them is empty. When one robot is stationary at a corner of the environment and the other robot moves along a wall (without losing visual contact) then a triangle of free space is mapped. By constructing an on-line triangulation of the free space the robots map the environment completely without any overlaps.

[4] The complete description of the algorithm is outside the scope of this paper (please refer to previous work[12, 11]).

The positional error is maintained low throughout the exploration by the use of cooperative localization. Figure 6a,b presents the pose estimates during the exploration when cooperative localization was used (marked as a "+") together with the position of the robot estimated using the recorded motion commands (marked as "*"); the map of the environment is shown. The left figure presents the trajectory of Robot 0 and the right figure presents the trajectory of Robot 1. Even though the actual trajectory of each robot was piecewise straight line and closely corresponds with the cooperative localization estimates, the motion commands show a systematic drift (marked as "*" in Figure 6a,b), the drift corresponds to the odometry error during the exploration.

Figure 7 presents experimental results from an indoor environment in the corridors of our building using two superscout robots. The pose *pdf* of each robot is plotted along the trajectory. At each step the set of particles has been spatially integrated and then added to the plot (the higher the peak the more accurate the pose estimate). Sub-figure 7a presents the trajectory of Robot 0 which is equipped with the laser range finder. Robot 1 is equipped with the three plane target, and the *pdf* of the robot's pose can be seen in Figure 7b. The resulting map is shown in Figure 6c.

4 Conclusions

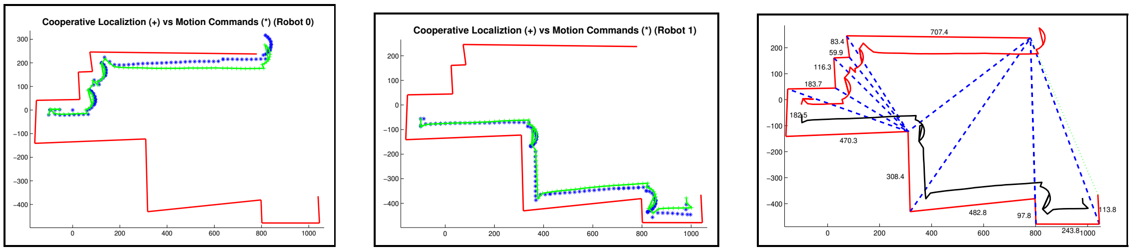
We have examined the details of a particle filter used to estimate the pose of a pair of robots during collaborative exploration. This detailed presentation has afforded us an opportunity to consider several subtleties that are typically omitted in papers on the subject. In fact, these design choices have a substantive bearing on the performance of such systems. Specifically, inappropriate choices of the number of particles, the frequency of resampling and the choice of the weighting function can lead to excessive computational burden (if the number of particles is too large, for example) or failure to accurately track the correct pose (if the number of particles is too small or if frequent resampling rises the variance of the particle population).

In ongoing work we are also considering more elaborate particle resampling methods that dynamically trade off efficiency for potential robustness. By estimating the parameters of the particle cloud, it seems possible to vary the model complexity on an as-needed basis. Further work involves the introduction of an additional weighting function based on other sensory input, such as the sensor used for wall following.

References

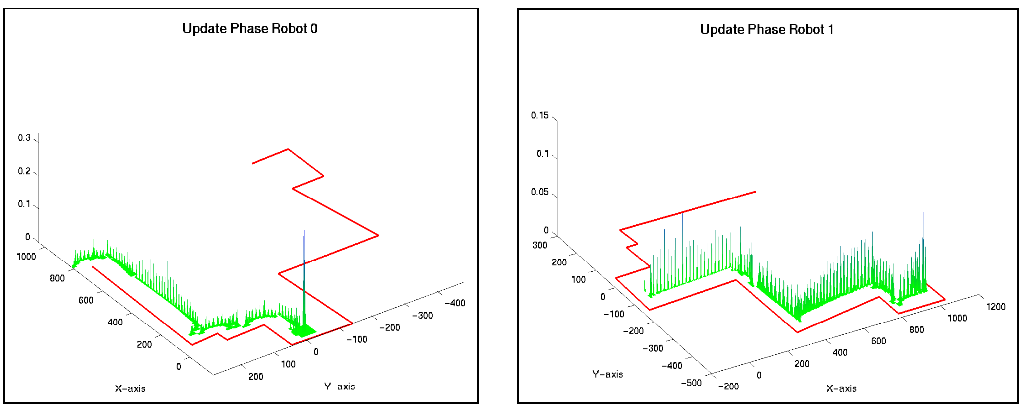
[1] M. Black and D. Fleet. Probabilistic detection and tracking of motion boundaries. *Int. J. of Computer Vision*, 38(3):231–245, 2000.

[2] J. Carpenter, P. Clifford, and P. Fearnhead. An improved particle filter for non-linear problems. *IEE proc. - Radar, Sonar and Navigation*, 146:2–7, 1999.



(a) (b) (c)

Figure 6: *(a)The cooperative localization estimates (*) versus the trajectory resulting from the motion commands (+) given to robot R0 during the exploration. (b) The same for robot R1. (c)The triangulation map produced using only the sonar for mapping. The trajectory of Robot 0 is marked in magenta and the trajectory of Robot 1 is marked black. The walls are the outer solid lines (lengths in cm). The internal diagonals that define the triangulation are marked as dashed lines.*



(a) (b)

Figure 7: *The two robots exploring a convex area. Both sub-figures present a spatial integration of the particles during the entire trajectory. Note that the height of each peak represents accuracy (high peak more accurate estimate) (a) The trajectory of Robot 0. (b) The trajectory of Robot 1.*

[3] F. Dellaert, W. Burgard, D. Fox, and S. Thrun. Using the condensation algorithm for robust, vision-based mobile robot localization. In *IEEE Conf. on Computer Vision and Pattern Recognition*, 1999.

[4] Arnaud Doucet, Nando De Freitas, and Neil Gordon. *Sequential Monte Carlo Methods in Practice*. Springer-Verlag, Series Statistics for Engineering and Information Science, January 2001.

[5] N.J. Gordon, D.J. Salmond, and A.F.M. Smith. Novel approach to nonlinear/non-gaussian bayesian state estimation. *IEE Proc. For Radar and Signal Processing*, 140(2):107 –113, April 1993.

[6] M. Isard and A. Blake. Condensation-conditional density propagation for visual tracking. *Int. J. of Computer Vision*, 29(1):2–28, 1998.

[7] Patric Jensfelt, Olle Wijk, David J. Austin, and Magnus Andersso. Feature based condensation for mobile robot localization. In *IEEE Int. Conf. on Robotics & Automation (ICRA)*, pages 2531–2537, 2000.

[8] John J. Leonard and Hugh F. Durrant-Whyte. Mobile robot localization by tracking geometric beacons. *IEEE Trans. on Robotics & Automation*, 7(3):376–382, June 1991.

[9] Jun S. Liu, Rong Chen, and Tanya Logvinenko. A theoretical framework for sequential importance sampling and resampling. In A. Doucet, N. de Freitas, and N.J. Gordon, editors, *Sequential Monte Carlo in Practice*. Springer-Verlag, January 2001.

[10] Ioannis Rekleitis, Robert Sim, Gregory Dudek, and Evangelos Milios. Collaborative exploration for the construction of visual maps. In *IEEE Int. Conf. on Intelligent Robots & Systems*, pg. 1269–1274, 2001.

[11] Ioannis Rekleitis, Gregory Dudek, and Evangelos Milios. Graph-based exploration using multiple robots. In *5th Int. Symposium on Distributed Autonomous Robotic Systems*, pg. 241–250, Oct. 2000.

[12] Ioannis M. Rekleitis, Gregory Dudek, and Evangelos Milios. Multi-robot collaboration for robust exploration. *Annals of Mathematics and Artificial Intelligence*, 31(1-4):7–40, 2001.

[13] Ioannis Rekleitis, Gregory Dudek, and Evangelos Milios. Multi-robot cooperative localization: A study of trade-offs between efficiency and accuracy. In *IEEE/RSJ/ Int. Conf. on Intel. Robots & Systems*, 2002.

[14] Stergios I. Roumeliotis and George A. Bekey. Collective localization: A distributed kalman filter approach to localization of groups of mobile robots. In *IEEE Int. Conf. on Robotics & Automation*, pg. 2958–2965, Apr. 2000.

[15] Nikos Vlasis, Bas Terwijn, and Ben Krose. Auxiliary particle filter robot localization from high-dimensional sensor observations. In *IEEE Int. Conf. in Robotics & Automation*, v. 1, pg. 7–12, May 2002.