Particle Attraction Localisation

Damien George
(dpgeorge@students.cs.mu.oz.au)

Nick Barnes
(nmb@cs.mu.oz.au)

Dept. Computer Science and Software Engineering
The University of Melbourne, Vic, 3010, AUSTRALIA

***Abstract*— In this paper we present an original method for Bayesian localisation based on particle approximation. Our method overcomes a majority of problems inherent in previous Kalman filter and Bayesian approaches, including the recent Monte Carlo Localisation methods. The algorithm converges quickly to any desired precision. It does not over-converge in the case of highly accurate sensor data and thus does not require a mixture-based approach. Also, the algorithm recovers well from random repositioning. These benefits are not hindered by computation which can be performed in real time on low powered processors. Further, the algorithm is intuitive and easy to implement. This algorithm is evaluated in simulation and has been applied to our entrant in the Sony Four Legged League of RoboCup, where it has been tested over many hours of international competition.**

I. Introduction

The pose of a mobile robot describes its location and orientation within an environment. Keeping track of pose is a general problem known as *localisation*. Traditional algorithms required the initial robot location to be known but there are an increasing number of problems which cannot be solved with this restriction. As an extension, *global localisation* algorithms have the ability to determine the initial pose of the robot and recover from drastic errors due to some external influence. The kidnapped robot problem, where the robot is picked up and placed in a random and unknown location, is considered a good example of an external influence and a good test for any global localisation algorithm.

A. Localisation Techniques

There are two main approaches to localisation. The traditional approach is based on the Kalman filter [1] where the state of a system is represented by a Gaussian function. The state is incrementally updated given odometric and landmark readings, also modelled by Gaussian functions. The Kalman filter has been successfully applied to many areas including localisation in robotics (e.g., [2]). The major limitation for localisation is the Kalman filter's inability to represent multiple solutions. Multiple hypothesis can occur when limited sensor readings do not yield enough information to arbitrate between two or more possible poses.

To overcome this limitation, extensions combine multiple Gaussian distributions (e.g., [3]) and are able to represent more than one hypothesised pose. The basic Kalman filter equations are used to integrate odometry and landmark information into a *set* of independent Gaussian representations. Each Gaussian is weighted by a probability indicating the relative confidence that the robot is at the associated location. Updating this set of Kalman filters can become computationally expensive with a large number of hypotheses. Since the environment and pose are represented by a Gaussian, the algorithm does not generalise to arbitrary distributions.

Most alternative techniques are Bayesian filters. They employ a probability distribution, known as the *prior* distribution, over the state space of robot pose. Given a sensor reading, Bayes theorem is applied to determine the *posterior* distribution. The difference between these techniques is the way the distribution is stored and maintained. Grid based methods (e.g., [4]) quantise the state space into cells with an associated probability. By its location in the grid, each cell implicitly represents a distinct and independent location in the state space and hypothesises the actual pose of the robot. For large areas with fine grained quanta this method is computationally slow and increasing the size of the cells reduces the resolution.

Particle filters (e.g., [5], [6]) and most notably Monte Carlo Localisation (MCL) [7], overcome efficiency problems by distributing particles over the approximated state space in a *non-uniform* way. Each particle represents a distinct pose by a state vector and a probability representing the relative confidence the robot is at this location. The important difference to the multiple Gaussian Kalman filter is that the state vector of a single particle is simpler and more efficient to update than a Gaussian function, allowing a larger number of particles to be used to approximate a general probability distribution. Unlike uniformly separated samples maintained with grid based methods, particle filters allow concentration of computation in areas where the robot is most likely to be.

While basic particle filters can globally localise a robot and efficiently track its motion, they are subject to over-convergence. The algorithm relocates particles with low probabilities to regions of state space populated by particles with the highest probabilities. In the extreme case, this leads to all particles occupying the same location and the algorithm ignores contradicting sensor readings.

In less extreme cases, the algorithm is biased towards sensor readings which confirm the current distribution. To overcome this problem [8] takes basic MCL, explicitly checks for over-convergence and resets, or re-samples, a certain number of particles. While this eliminates the more extreme cases of over-convergence, the algorithm still has problems when the hypothesis is close to, but does not precisely predict the correct location.

The idea of re-sampling particles from the distribution given by the sensor readings is improved upon in Mixture-MCL [9], [10]. The dual of MCL takes the state vectors of the particles from the distribution given by the sensor readings. The probabilities of the particles are then adjusted in proportion to the prior distribution. Although dual MCL does not suffer from over-convergence, it is highly susceptible to noise. Mixture-MCL combines ordinary MCL and dual MCL by randomly applying one or the other algorithm to each particle. While this approach achieves accuracy, noise immunity and is not subject to over-convergence, dual MCL is complicated to implement and Mixture-MCL requires substantially more computational resources than the basic particle filters.

B. Particle Attraction Localisation

In this paper we present a simpler and original algorithm based on particles: Particle Attraction Localisation (PAL). All the traditional problems of globally locating and tracking a robot are solved and it does not suffer from over-convergence. Also, it yields a uniform distribution over non-point regions in hypothesis space. The accuracy of PAL is proportional to that of sensor readings while still retaining the ability to react to both small errors in location and random repositioning. Furthermore, PAL is an intuitive algorithm, easy to implement and computationally efficient. The following sections describe the model in general mathematical terms, give a specific implementation of PAL and discuss detailed simulation results and experiments with a Sony Aibo robot.

II. Mathematical Model

Bayes theorem relates conditional probabilities:

$$p(\boldsymbol{x}|\chi) = \frac{p(\boldsymbol{x})}{p(\chi)}\, p(\chi|\boldsymbol{x}), \tag{1}$$

where $\boldsymbol{x}$ and χ are probabilities of an event. For localisation, we take $\boldsymbol{x}$ to be a proposed robot pose and χ an observation. Also:

- $p(\boldsymbol{x})$ is the prior probability of the pose $\boldsymbol{x}$;
- $p(\chi)$ is a normalisation constant;
- $p(\chi|\boldsymbol{x})$ is the probability of observation χ given the robot is at pose $\boldsymbol{x}$ (sensor model); and,
- $p(\boldsymbol{x}|\chi)$ is the posterior probability of the original pose $\boldsymbol{x}$.

The application of (1) in an iterative Bayesian filter takes the posterior probability of one iteration to be the prior probability of the next. This iterative use requires successive observations to be independent of one another — the *Markov* assumption. For a more rigorous treatment and derivation of the Bayes filter see [10].

A. Particle Representation

A generalised robot pose, including location and orientation, can be represented by an m dimensional state vector. For example, two spatial components and an orientation (measured counterclockwise from the positive x-axis), (x, y, ϕ). We use the notation $||\boldsymbol{x}_i, \boldsymbol{x}_j||$ as the distance between vectors $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$.

PAL aims to approximate the robot pose in this space with n particles. Each particle is represented by a state vector $\boldsymbol{x}_i = (x_{i1}, x_{i2}, \ldots, x_{im})$ with an associated probability p_i. Mathematically, we write the distribution as a set of combined state vectors and probabilities (particles): $D = \{\langle \boldsymbol{x}_i, p_i \rangle \mid i = 1 \ldots n\}$. Each state vector in the set D represents a possible robot pose. The probability reflects the confidence the robot has this pose. For the n particles to produce a consistent distribution, $\sum_{i=1}^{n} p_i = 1$.

B. Update

When new pose information is obtained, it is incorporated into PAL as either static or dynamic information. Static information forces a *constraint* on the possible pose. Dynamic information (e.g., odometry) refers to a *displacement* within state space. For both static and dynamic information, each particle is updated individually by a transformation, consistent with the sensor reading and takes the particle to a new position in the state space.

Along with static and dynamic transformations, the distribution is also modified by neighbour repulsion. This attempts to keep particles spread around the most relevant regions in state space, explicitly avoiding over-convergence. Repulsion also allows particles to diffuse evenly over state space when few or no sensor readings are made. This is an advantage over standard particle filters which rely on random sampling.

The transformations are finalised with boundary checking and normalisation, ensuring the validity of the posterior distribution.

By updating in three phases, it is possible to apply an update only when necessary, leading to efficient use of computation. For example, if odometric readings are available often but sensor readings are rare, then the dynamic update transformation will be applied more often than the static update. Note that there is no restriction on the order of the applied transformations.

If $\mathcal{T}$ is a transformation, t the current iteration and $t+1$ the next, then D is updated:

$$D(t+1) = \mathcal{T}(D(t)). \tag{2}$$

1) Static Information: The basic idea is to attract each particle to the closest pose in state space consistent with the static information. Bayes theorem is then applied to update the confidence of each particle. For maximum accuracy and convergence speed, static information should be incorporated as often as it is available, although this is not a strict requirement.

A given sensor reading χ is represented as a constraint $s_\chi(\boldsymbol{x}) = 0$ on a state vector $\boldsymbol{x}$. The form of s_χ can range from a precise location to a set of independent parameterised locations. For example, if our robot is facing the only elevator in the building, then it is at a unique location. The constraint would be $s_\chi(x, y, \phi) = (x, y, \phi) - (l_x, l_y, l_\phi + \pi)$ where (l_x, l_y, l_ϕ) is the location of the elevator. The distance and bearing to a wall would lead to a constraint with a continuum of solutions.

Given a constraint s_χ, we determine a new intermediate state vector for each particle denoted $\boldsymbol{x}'_i(t+1)$. This pose must satisfy the constraint and be the closest pose to the previous $\boldsymbol{x}_i(t)$ as defined by the distance metric. Formally:

- $s_\chi(\boldsymbol{x}'_i(t+1)) = 0$; and,
- $||\boldsymbol{x}'_i(t+1),\ \boldsymbol{x}_i(t)||$ is minimised.

For the elevator example, there is only one solution, thus $\boldsymbol{x}'_i(t+1) = (l_x, l_y, l_\phi + \pi)$. The requirement that the distance to the old pose be minimised is necessary only to decide among multiple solutions. If the robot detected a wall, then $\boldsymbol{x}'_i(t+1)$ would be equated with the location of the wall closest to the old pose $\boldsymbol{x}_i(t)$. After determining $\boldsymbol{x}'_i(t+1)$ for each particle, its pose is revised to a weighted average of the original pose and this intermediate pose:

$$\boldsymbol{x}_i(t+1) = \boldsymbol{A}\boldsymbol{x}'_i(t+1) + (\boldsymbol{I} - \boldsymbol{A})\,\boldsymbol{x}_i(t), \qquad (3)$$

where $\boldsymbol{I}$ is an identity matrix and $\boldsymbol{A}$ a diagonal $m \times m$ damping matrix. The entries of $\boldsymbol{A}$ lie between zero and one inclusive and depend on the environment. They allow different damping in each dimension of state space and prevent particles responding too rapidly to sensor noise. Noise can be reduced further by making damping dependent on the confidence of a particle — more confident particles move less.

Bayes theorem in the form of (1) is applied to update the confidence of each particle (α is a normalisation constant):

$$p_i(t+1) = \alpha\, p_i(t)\, p\,(\chi|\boldsymbol{x}_i(t)) \qquad (4)$$

2) Dynamic Information: Dynamic information specifies a discrete change in robot pose. Conceptually, the pose of each particle is updated as though that particle had moved in such a way as to produce the given information. Dynamic information is commonly specified in local robot space, $\boldsymbol{d}$, and thus requires a transformation to the global space for each particle, $\boldsymbol{d}'_i$. The update transformation for a particle is then:

$$\boldsymbol{x}_i(t+1) = \boldsymbol{x}_i(t) + \boldsymbol{d}'_i \qquad (5)$$

For example, consider an odometric reading that indicates our robot has moved forward a distance f (in its own coordinate system). Each particle has a local coordinate frame defined by ϕ_i. A particle facing along the positive x-axis will be updated by increasing x_i by f. No change is made to the probability values of the particles. They are all updated equally and there is no reason for one to be more confident due to motion. A constant factor modelling a decrease in confidence would just be normalised out. Some Bayesian localisation methods (e.g., [4], [7], [9], [10]) blur the distribution when incorporating odometric reading. Our method makes up for this with neighbour repulsion.

3) Neighbour Repulsion: Using only the preceding two update methods leads to all particles converging to the same state vector — the proposed robot location. With all particles in the same state, they all behave in the same way, leading to a distribution of effectively one particle. This problem is known as over-convergence and [10] details one way of preventing it. With neighbour repulsion, we propose an alternative. The main aim is to keep the particles a certain distance from each other by repelling their closest neighbours. There are two parameters:

- $\lambda > 0$: characteristic distance between two neighbours. A larger value leads to larger separation.
- $\eta \geq 0$: repulsion power per iteration. A larger value results in faster spreading of particles.

Given an initial distance r between two neighbours, we determine the desired increase in their distance, Δr, as:

$$\Delta r = \eta e^{-r/\lambda} \qquad (6)$$

Note that Δr is non-negative and falls rapidly as r becomes greater than λ.

Since no external information used in the neighbour repulsion update phase, it is important that it does not change the proposed robot location. The maximum remains invariant under neighbour update if the particle with the largest p_i is skipped when repelling neighbours. To keep the average of the entire distribution invariant it is sufficient to ensure a pair of particles keeps the same average.

$$\begin{aligned} r_i &= \frac{\Delta r}{1 + p_i/p_j} \\ r_j &= \frac{\Delta r}{1 + p_j/p_i} \end{aligned} \qquad (7)$$

The magnitude by which each particle is repelled is specified by (7), with direction along the line joining the two particles. If $\hat{\boldsymbol{n}}_{ij}$ is a unit vector pointing from particle i to particle j then the repulsion is:

$$\begin{aligned} \boldsymbol{x}_i(t+1) &= \boldsymbol{x}_i(t) + r_i\hat{\boldsymbol{n}}_{ji} \\ \boldsymbol{x}_j(t+1) &= \boldsymbol{x}_j(t) + r_j\hat{\boldsymbol{n}}_{ij} \end{aligned} \qquad (8)$$

Again, no change is made to the individual particle probabilities. The distribution is kept consistent with the scaling r_i and r_j dependent on the probabilities of the two neighbours.

The neighbour repulsion update should be applied at a constant rate, reflecting the temporal nature of the repulsion. This also facilitates the adjustment of the parameters λ and η, independent of the number of sensor readings.

4) Update Finalisation: After an update transformation, a particle state may become invalid due to a physical obstacle or the edge of state space. In both cases the posterior probability of the particle, $p_i(t+1)$, is set to a small number, ϵ, ensuring the particle does not influence the hypothesised robot pose. If the state $\boldsymbol{x}(t+1)$ represents the location of some obstacle then it is left as is. This allows particles to "move through walls" and find the correct robot location. If $\boldsymbol{x}(t+1)$ is found to lie outside the environment, it is brought back to the nearest valid location. After checking for validity, probabilities are normalised and the new set of particles $D(t+1)$ represented by $p_i(t+1)$ and $\boldsymbol{x}_i(t+1)$ replaces the old set $D(t)$, as defined by (2).

III. Implementation

We augment the general algorithm presented above with an implementation for a robot in a two dimensional plane with an orientation (x, y, ϕ). Each particle is specified with a state vector $\boldsymbol{x}_i = (x_i, y_i, \phi_i)$ and a probability p_i.

The distribution of the particles maintained by PAL is an estimate of robot pose. However, in some situations we may be forced to give a point estimation of robot location. We consider two simple approaches: *maximum* – take the state $\boldsymbol{x}_i$ of the particle with the largest p_i; and *average* – take the weighted average over all the particles with $\sum_{i=1}^{n} p_i \boldsymbol{x}_i$. Both methods predict the same location when the particles have converged. If the distribution has multiple hypotheses, the average will generally give a result at none of these. The maximum will choose the most probable location and although it might be incorrect, it is the best that can be done in an ambiguous situation.

A. Landmarks

We develop a static update procedure for a landmark with a known global location (l_x, l_y). We assume the landmark sensor determines the relative range R and bearing ψ from the robot. The information from a sensor is then a 4-tuple: (l_x, l_y, R, ψ). In 3-dimensional state space this produces a helical constraint which we approximate with independent constraints on (x_i, y_i) and ϕ_i (θ_i is the angle the particle makes with the x-axis and the landmark):

$$\begin{aligned} x_i^2 + y_i^2 - R^2 &= 0 \\ \phi_i - (\theta_i + \pi - \psi) &= 0 \end{aligned} \tag{9}$$

Using these constraint equations, we now determine the intermediate state vector for each particle, $\boldsymbol{x}'_i(t+1)$. This satisfies the given constraints and is close to $\boldsymbol{x}_i(t)$. The explicit components are:

$$\begin{aligned} x'_i(t+1) &= l_x + R\cos\theta \\ y'_i(t+1) &= l_y + R\sin\theta \\ \phi'_i(t+1) &= \theta_i + \pi - \psi \end{aligned} \tag{10}$$

We introduce parameters a_r and a_ϕ for radial and angular damping respectively. Then substituting (10) into (3):

$$\begin{aligned} x_i(t+1) &= a_r(l_x + R\cos\theta) + (1-a_r)x_i(t) \\ y_i(t+1) &= a_r(l_y + R\sin\theta) + (1-a_r)y_i(t) \\ \phi_i(t+1) &= a_\phi(\theta_i + \pi - \psi) + (1-a_\phi)\phi_i(t) \end{aligned} \tag{11}$$

Applying (11) to each particle attracts the distribution to the observed landmark. The posterior probabilities are found using (4).

B. Odometry

A local robot space odometric reading $\boldsymbol{d} = (d_x, d_y, d_\phi)$ is transformed to global space for an individual particle by a rotation $\phi_i(t)$:

$$\begin{aligned} d'_{ix} &= d_x\cos(\phi_i(t)) - d_y\sin(\phi_i(t)) \\ d'_{iy} &= d_x\sin(\phi_i(t)) + d_y\cos(\phi_i(t)) \\ d'_{i\phi} &= d_\phi \end{aligned} \tag{12}$$

The pose of each particle is updated by application of (5) using $\boldsymbol{d}'_i$ defined by (12). This transformation is relative to the individual particle orientation and will generally lead to different particles moving in different directions for the same $\boldsymbol{d}$. This is the desired behaviour since the distribution has the capacity to represent more than one independent possible robot pose.

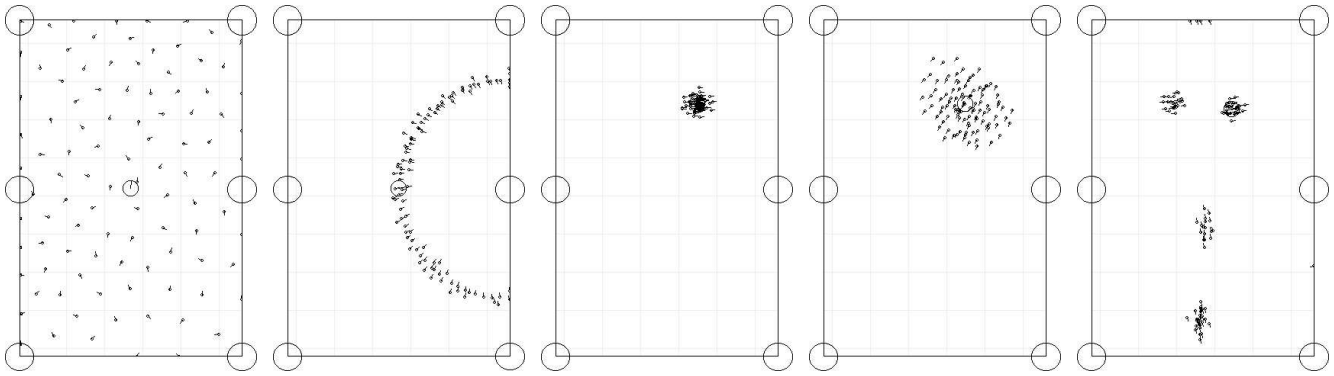
C. Neighbour Repulsion

Implementation of neighbour repulsion is relatively straight forward. Ideally, repulsion should be computed for every possible pair of particles, an $O(n^2)$ computation. In order to reduce this, each particle has a small number of exclusive neighbours with which it performs the repulsion update. Each particle also maintains an iterator which cycles through those particles not in its exclusive neighbour list. When the iterator comes to a particle which is closer than one in the neighbour list a replacement is made. This method aims to keep computation of neighbour repulsion to the particles which are closest to each other.

IV. Experimental Results

The performance of PAL was tested extensively in simulation using ArSim[1] and on a Sony Aibo robot. PAL

[1] ArSim is available at http://www.cs.mu.oz.au/robocup.



(a) (b) (c) (d) (e)

Fig. 1. Actual results after a number of iterations showing robot (large circle), particles (small dots) and landmarks (edge circles). (a) Initially the particles are evenly distributed reflecting the complete lack of knowledge of the location. (b) After observing the middle right landmark, the particles gather in an arc. (c) Two landmarks have been observed, uniquely locating the robot. (d) The robot does not make any observations for a while and the particles spread out. (e) Representation of multiple hypotheses in a symmetric environment. Given that diagonally opposite beacons are the same, PAL handles this situation well.

was implemented from the equations derived in Section III and the results are presented in the following sections.

A. Simulation

ArSim simulates a differential drive robot with odometry and a landmark detector. The environment was modelled after the standard Sony Four Legged League RoboCup field with six unique landmarks. The landmark detector has a 120° field of view and a uniformly distributed noise model in bearing and range. Odometry has a similar noise model.

The simulated robot was driven around the field, and observed and processed landmark readings and odometry every iteration. Fig. 1 shows a series of instances of PAL localising the robot. The gathering of particles in an arc (Fig. 1b) is desirable, since only one beacon is visible. Furthermore, the particles are evenly distributed along this arc. In the Sony Four Legged League the landmarks are unique and so the correct robot location can be deduced by observing two landmarks. In contrast, Fig. 1e shows PAL representing multiple hypotheses when there is a large amount of symmetry in the environment.

Fig. 2 shows the ability of PAL to promptly recover from the kidnapped robot situation and converge to a low error (measured as Cartesian distance between actual and estimated location). Note that both before and after the kidnapping the sensors on the robot had two landmarks in range, allowing unique identification of location.

Fig. 3a shows that an *increase* in the number of particles leads to a *decrease* in the convergence time — consistent with alternative particle filters. Fig. 3b shows the effect of radial damping on convergence time and hypothesis error. With a larger damping factor there is a larger average hypothesis error but the convergence time is less.

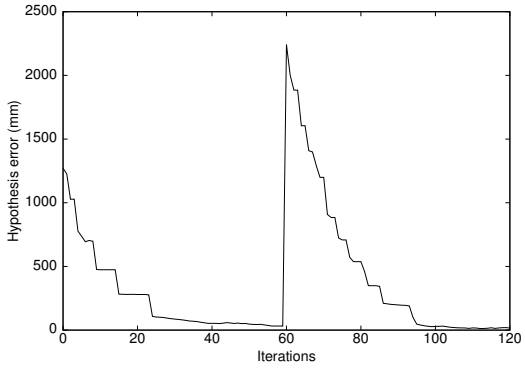


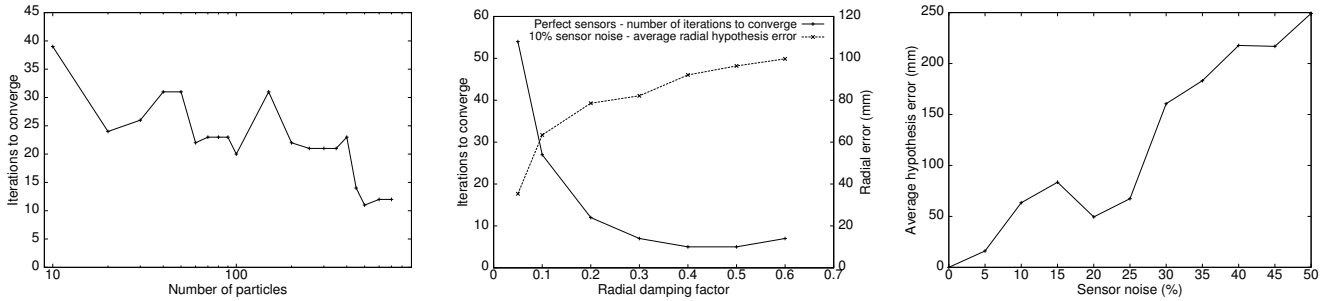
Fig. 2. The kidnapped robot problem. After 60 iterations the robot is transported to an unknown location. The PAL algorithm recovers promptly.

Thus convergence speed is traded for average error when selecting the damping factor.

The average error in the hypothesis increases with an increase in sensor noise, as shown in Fig. 3c. While the error is not monotonic as sensor noise increases, it does show a linear relationship in contrast to the exponential increase in [10].

B. Legged Robots

PAL was applied to a Sony Aibo ERS-210 as used in the Sony Four Legged League RoboCup competition in 2002. We claim that the application of the PAL algorithm in this domain was competitive and in many cases superior to the opposing implementations of Kalman filters and MCL algorithms. The algorithm performed flawlessly in countless hours of practice matches and competition, placing our team fourth out of nineteen in the world.



(a) (b) (c)

Fig. 3. (a) Iterations for hypothesis to converge within a small radius of actual location versus number of particles. (b) Effect of radial damping on convergence speed (solid curve) and average hypothesis error (dashed curve). (c) Average radial error versus sensor noise.

Odometry from the robot was calculated by counting steps and multiplying by the length of each step. Sensor readings are taken from visual identification of landmarks, for which distance estimation has up to 30% error for long range landmarks. The robot was programmed to move to 17 known positions using localisation and wait while the actual position was measured. The field is 2.9 meters by 4.4 meters and we obtained a 5.8% error in horizontal and 2.5% error in vertical position. There was no measurable error in bearing.

V. Conclusion

In this article we presented an original method for general global robot localisation called Particle Attraction Localisation. A probability distribution of the robot pose is stored as a set of particles. The particles are updated in three distinct ways:

- **Static information** provides a constraint on the state space which attracts the particles and updates their probabilities according to Bayes theorem.
- **Dynamic information** moves each particle through the state space as specified by the sensors.
- **Neighbour repulsion** keeps the distribution spread evenly about the hypotheses to prevent over-convergence and cope with ambiguous situations.

After an update, boundary checking and normalisation are applied to the posterior distribution to ensure its validity. These procedures can be carried out in a straight forward and efficient manner leading to high precision localisation.

A full implementation of the abstract model was given for a robot in 3-dimensional state space with position vector (x, y, ϕ). This was then shown to behave correctly in a simulated environment and in application to a Sony Aibo robot in the four-legged league of RoboCup.

This new algorithm shows some promising features and there are many areas in which research can continue, including improvements in response to false positive sensor readings. We will also examine the performance of PAL for visual tracking problems.

VI. References

[1] R. Kalman. A New Approach to Linear Filtering and Prediction Problems. *Jour. Basic Engineering*, vol. 82-D, 1960.

[2] H. Durrant-Whyte. An Autonomous Guided Vehicle for Cargo Handling Applications. *Int. Jour. Robotics Research*, vol. 15, no. 5, 1996.

[3] P. Jensfelt and S. Kristensen. Active global localization for a mobile robot using multiple hypothesis tracking *IEEE Trans. on Robotics and Automation*, vol. 17, no. 5, 2001. Pages: 748- 760.

[4] P. Buschka, A. Saffiotti and Z. Wasik. Fuzzy Landmark-Based Localization for a Legged Robot. *Proc. of IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, 2000.

[5] P. Jensfelt, D. Austin, O. Wijk and M. Andersson. Feature Based Condensation for Mobile Robot Localization. *IEEE Int. Conf. on Robotics and Automation*, 2000.

[6] P. Jensfelt, O. Wijk, D. Austin and M. Andersson. Experiments on Augmenting Condensation for Mobile Robot Localization. *IEEE Int. Conf. on Robotics and Automation*, 2000.

[7] D. Fox, W. Burgard, F. Dellaert and S. Thrun. Monte Carlo Localization: Efficient Position Estimation for Mobile Robots. *Proc. of the 16th Nat. Conf. on Artificial Intelligence*, 1999.

[8] S. Lenser and M. Veloso. Sensor Resetting Localization for Poorly Modelled Mobile Robots. *Proc. IEEE Int. Conf. on Robotics and Automation*, 2000

[9] S. Thrun, D. Fox and W. Burgard. Monte Carlo Localization With Mixture Proposal Distribution. *Proc. Nat. Conf. on Artificial Intelligence*, 2000. *AAAI*, 2000.

[10] S. Thrun, D. Fox, W. Burgard and F. Dellaert. Robust Monte Carlo Localization for Mobile Robots. *Artificial Intelligence*, vol. 128, no. 1-2, 2001.