

A Comparative Analysis of Particle Filter based Localization Methods*

Luca Marchetti, Giorgio Grisetti, Luca Iocchi

Dipartimento di Informatica e Sistemistica
Università “La Sapienza”, Rome, Italy
Via Salaria 113 00198 Rome Italy
E-mail: <firstname.lastname>@dis.uniroma1.it

Abstract

The knowledge of the pose and the orientation of a mobile robot in its operating environment is of utmost importance for an autonomous robot. Techniques solving this problem are referred to as self-localization algorithms. Self-localization is a deeply investigated field in mobile robotics, and many effective solutions have been proposed. In this context, Monte Carlo Localization (MCL) is one of the most popular approaches, and represents a good tradeoff between robustness and accuracy. The basic underlying principle of this family of approaches is using a Particle Filter for tracking a probability distribution of the possible robot poses.

Whereas the general particle filter framework specifies the sequence of operations that should be performed, it leaves open several choices including the observation and the motion model and it does not directly address the problem of robot kidnapping.

The goal of this paper is to provide a systematic analysis of Particle Filter Localization methods, considering the different observation models which can be used in the RoboCup soccer environments. Moreover, we investigate the use of two different particle filtering strategies: the well known Sample Importance Resampling (SIR) filter, and the Auxiliary Variable Particle filter (APF).

The results of the experiments presented in this work show how the localization’s performances are affected by the choices in the Particle Filter implementation, and aims to provide additional guidelines in developing Particle Filter based algorithms.

1 Introduction

The knowledge of the pose and the orientation of a mobile robot in its operating environment is of utmost importance for an autonomous robot. Self-localization is a well known problem in mobile robotics, and many effective solutions have been proposed.

The presence of an initial position guess strongly condition the development of a localization algorithm. Whenever a position guess is available, a localization technique has to keep consistent the estimates of the system over time, but it has not to determine the robot location from scratch. This family of techniques goes under the name of *position tracking*.

Conversely, when an initial position guess is not available, the system has to disambiguate between the potentially many different pose hypotheses resulting from the observation matching. Once the correct position is assessed, the system has to keep such an estimate consistent. This approaches goes under the name of *global localization*.

The prototype of algorithms proposed to solve position tracking is Kalman Filter localization [11], while global positioning encloses common frameworks like Multi Hypotheses Localization [7], Histogram Filters [1] and Particle Filters [4].

*This is an extended version of the RoboCup Symposium 2006 paper.

In the last years Particle Filter Localization, also known as Monte Carlo Localization (MCL), became one of the most popular approaches for solving the global localization problem. The general framework has been developed for both feature based maps [9] and grid based maps [2].

Whereas the implementation of a particle filter for localization is straightforward, its performances are strongly affected by the modeling of the process to estimate. Namely the user has to specify the system *motion model*, that is the probability distribution of successor states conditioned to the odometry readings, and the *observation model* that describes the likelihood of a given observation given the current robot position.

In the RoboCup Four-legged league the localization problem becomes a challenging task, because of the following reasons:

- The only sensor that can be used for acquiring measures of the environment is a low resolution camera. Additionally, the images are corrupted by the fast motion of the robot camera with respect to the mobile base.
- The robot motion is affected by a considerable amount of noise due to both the presence of opponents in the field of play and to the poor accuracy of the odometry.
- The computational power available for localization is rather limited.

The dynamic environment strongly violates the Markov assumption which underlies most of the approaches proposed in literature. In order to cope with such violations, several extensions have been proposed to the original Bayes formulation of the localization problem. To this end the most popular technique is known as *sensor resetting* [10]. It consists in bootstrapping the estimator with hypotheses based on the raw observations.

The goal of this paper is to present a systematic analysis performed on the Particle Filter localization, when considering the different observation models which can be used in the RoboCup Four Legged league contexts. Moreover, we investigate the use of two different particle filtering strategies: the well known Sample Importance Resampling (SIR) filter [5], and the Auxiliary Variable Particle filter (APF) [13].

Localization based on APF has been previously proposed by Vlassis *et al.* [15] for solving the vision based localization problem, together with a nonparametric estimate of the likelihood function. The main focus of their work is on how to compute a satisfactory nonparametric estimate of the direct observation model $p(x|z)$, expressing the probability of being in a given location x given the observed panoramic camera image z . Such a distribution is expressed through a Gaussian mixture learned from the data.

In contrast to [15], the contribution of our work is to investigate possible variants of the particle based localization algorithms, for parametric (feature based) observation models, treating APF as an additional degree of freedom.

The rich literature about self localization has been organized by Gutmann and Fox [6]. In this work EKF, histogram and particle filter localization approaches have been compared under different settings, in the RoboCup legged environment. Subsequently, Kristensen and Jensfelt [8] extended the comparison with their Multi Hypotheses Localization (MHL) framework, a multi-modal extension to the EKF.

Whereas the MHL localization has been proved to be extremely efficient when the environment can be modeled by landmarks, its implementation is rather complex if compared with a the common particle based approaches.

The focus of this work is to provide the reader with some guideline on how to practically implement a particle filter localization algorithm. The different choices that can be made in the development of an MCL algorithm are investigated through a set of experiments, in which the details of the landmark based observation models and motion models for the RoboCup environment are analyzed.

2 Particle Filter

In this section we recall the Bayesian formulation of a state estimation problem and we derive the general recursive solution to obtain the target distribution. Then we particularize the formula to particle filters, and we consider two different resampling strategies.

2.1 Bayesian estimation

We denote with x the system state and with z the sensor data. The odometry is modeled as a system input u . With these conventions the target distribution to estimate is

$$p(x_t|z_{0:t}, u_{0:t}) \quad (1)$$

If the Markov assumption holds, namely if the current measurements are independent from the past ones given the current state, Eq. 1 can be estimated in a recursive way, as follows:

$$\begin{aligned} p(x_t|z_{0:t}, u_{0:t}) &= \frac{p(z_t|x_t) \int p(x_t|x_{t-1}, u_t) p(x_{t-1}|z_{0:t-1}, u_{0:t-1}) dx_{t-1}}{\int p(z_t|x_t) \int \underbrace{(p(x_t|x_{t-1}, u_t) p(x_{t-1}|z_{0:t-1}, u_{0:t-1}))}_{1/\eta} dx_{t-1} dx_t} \\ &= \eta p(z_t|x_t) \int p(x_t|x_{t-1}, u_t) p(x_{t-1}|z_{0:t-1}, u_{0:t-1}) dx_{t-1}. \end{aligned} \quad (2)$$

Here the normalization factor $1/\eta$ does not depend on x_t , therefore in the following we focus only on the computation of Eq. 2.

The evaluation is typically performed in two steps: the prediction step, which computes the distribution of the current state after integrating the odometry measurement u_t , and the update step which integrates the last observation in the predicted distribution. In the following we describe in detail these two operations

- In the **prediction Step**, the integral part of Eq. 2 is computed:

$$p(x_t|z_{0:t-1}, u_{0:t}) = \int \underbrace{p(x_t|x_{t-1}, u_t)}_{\text{motion model}} \underbrace{p(x_{t-1}|z_{0:t-1}, u_{0:t-1})}_{\text{prior}} dx_{t-1}$$

This is done by convolving the previous state probability distribution, the **prior**, with the so called robot **motion model** $p(x_t|x_{t-1}, u_t)$. While the prior comes from the previous step estimate, the motion model is a function which estimates the probability of leading in the state x_t given that the robot started from x_{t-1} and sensed the motion u_t through the odometry. The distribution resulting from a prediction step is, in general less informative than the prior distribution. This behavior captures the decrease in information resulting from performing a blind move.

- The **update step** computes the predicted distribution after incorporating the measurement z_t :

$$p(x_t|z_{0:t}, u_{0:t}) \propto \underbrace{p(z_t|x_t)}_{\text{observation model}} \cdot p(x_t|z_{0:t-1}, u_{0:t}) \quad (3)$$

This is done using the so-called **observation model**, a function that measures the likelihood of the current measurement z_t if the robot is in the pose x_t .

According to Eq. 2 for estimating the robot pose two functions should be known: the **motion model** and the **observation model**. These functions are characteristic of the system and they strongly influence the behavior of Bayes Localization. Depending on the assumptions made on the motion model, the sensor model and the estimated distribution, the computation of Eq. 3 leads to different algorithms like the Kalman Filter, the Histogram Filters and the Particle Filters. Due to the partial system observability, in approaching the localization problem it is often required to track multi-modal distributions. Additionally, the limited computational resources available makes it challenging to estimate the robot pose in a limited amount of time.

2.2 Particle filtering

One of the most popular algorithms used in localization is the so-called Monte Carlo Localization introduced by Dellaert *et al.* [4]. The core idea of the algorithm is to estimate a robot pose distribution

using a particle filter. The system state is represented through a set of samples in the robot pose space $\{x^{(i)} = (\mathbf{x}, \mathbf{y}, \theta)^{(i)}\} \in \mathbb{R}^2 \times [0 \dots 2\pi)$.

$$p(x) \simeq \frac{1}{N} \sum_{i=0}^N \delta_{x^{(i)}}(x)$$

here $\delta_{x^{(i)}}(\cdot)$ is the impulse function centered in the sample $x^{(i)}$. The denser are the samples in a state space region the higher the probability that the robot is in that region.

Ideally one wants to sample from the posterior distribution

$$x_t^{(i)} \sim p(x_t | z_{0:t}, u_{0:t})$$

but this is not possible in the general case because such a distribution is not available in closed form. However by representing the distribution through a set of weighted samples $\langle w^{(i)}, x^{(i)} \rangle$ it is still possible to carry on the estimate. The samples are drawn from a so called proposal distribution $q(x_t | z_{0:t}, u_{0:t})$, while the weights for each sample are computed according to the Importance Sampling Principle

$$w_t^{(i)} = \frac{p(x_t^{(i)} | z_{0:t}, u_{0:t})}{q(x_t^{(i)} | z_{0:t}, u_{0:t})} \quad (4)$$

By choosing the motion model $p(x_t | x_{t-1}^{(i)}, u_t)$ as proposal distribution, we can recursively compute the weights as

$$\begin{aligned} w_t^{(i)} &= \frac{p(x_t^{(i)} | z_{0:t}, u_{0:t})}{p(x_t | x_{t-1}^{(i)}, u_t)} \\ &\propto \frac{p(z_t | x_t^{(i)}) p(x_t^{(i)} | x_{t-1}^{(i)}, u_t)}{p(x_t^{(i)} | x_{t-1}^{(i)}, u_t)} w_{t-1}^{(i)} \\ &= p(z_t | x_t^{(i)}) w_{t-1}^{(i)}. \end{aligned}$$

It is worth to notice that, if the samples are generated from a sub-optimal proposal, the weights of most of them will approach zero after a few iterations. Therefore the filter spends a considerable amount of computational time in updating unlikely samples. This can be avoided by the resampling step, which consists in replacing the unlikely samples with the more likely ones.

In the following sections we present two particle filtering techniques which can be used for performing self localization: the Sampling Importance Resampling and the Auxiliary Variable Particle Filter algorithms.

2.3 Sampling Importance Resampling Filter

The three steps described above:

- sampling from a proposal distribution,
- evaluating the weights according to the importance function,
- resampling

can be found in the popular Sampling Importance Resampling (SIR) framework introduced by Gordon *et al.* [5], illustrated in Table 1.

One of the main problems of the SIR algorithm is the *degeneracy problem* [13]. If the ratio between the variance of the proposal distribution and the observation model is high, it can happen that only a few samples generated in the sampling steps have a meaningful weight. The subsequent application of the resampling operation results in the suppression of most of the samples generated, because only those with greater weight are replicated, replacing the low-weighted particles. This fact reduces the chances that the filter achieves to a correct convergence because it impoverishes the set diversity.

Algorithm 1 SIR particle filter

```

{ $x_t^{(i)}, w_t^{(i)}$ } $_{i=1}^N = SIR[\{x_{t-1}^{(i)}, w_{t-1}^{(i)}\}_{i=1}^N, z_t, u_t]$ 
FOR  $i = 1 : N$ 

    evolve samples using odometry
     $\tilde{x}_t^{(i)} \sim p(x_t | x_{t-1}^{(i)}, u_t)$ 
    calculate weight using observations
     $\tilde{w}_t^{(i)} = p(z_t | \tilde{x}_t^{(i)})$ 

ENDFOR
FOR  $i = 1 : N$ 

    normalize weights of samples
     $\tilde{w}_t = NORMALIZE[\tilde{w}_t^{(i)}]$ 

ENDFOR
{ $x_t^{(i)}, w_t^{(i)}$ } $_{i=1}^N = RESAMPLE[\{\tilde{x}_t^{(i)}, \tilde{w}_t^{(i)}\}_{i=1}^N]$ 

```

Table 1: Pseudo-code of a generic SIR Particle Filter

2.4 Auxiliary Particle Filter

Alternative filtering schemes like the Auxiliary variable particle filter have been introduced to lessen the degeneracy problem. The key idea of this algorithm is to select the samples that will be propagated in the subsequent updated estimate. Such a selection is performed by evolving the current filter state using a reduced noise motion model, and evaluating the sample weights according to the Importance Sampling (IS) principle [3]. Then a set of indices is sampled from the weights distribution, and only the surviving particles will be used for computing the updated particle generation, by means of the original motion model. This ensures an increased particle diversity, since the resampling is performed before evolving the filter.

More in detail the APF exploits the following factorization over the joint posterior of particle indices and robot poses

$$p(x_t, i | z_{0:t}, u_{0:t}) \propto p(z_t | \mu_t^{(i)}) p(x_t | x_{t-1}^{(i)}, u_t).$$

Here μ_t is a mean, a mode or some other indicator of the predicted distribution, designed so that

$$p(i | z_{0:t}, u_{0:t}) \propto p(z_t | \mu_t^{(i)}).$$

Under the above hypotheses, we can sample from $p(x_t, i | z_{0:t}, u_{0:t})$ by sampling an index j with probability $\lambda^{(j)} = p(z_t | \mu_t^{(j)})$. We denote with j^i that the value of the j^{th} sampled index is the original index i . Subsequently, we can sample from the motion model $x_t^{(j)} \sim p(x_t | x_{t-1}^{(j^i)})$, according to the value of the state referred to by the drawn index.

According to the IS principle, the resulting weights will be:

$$w^{(j)} = \frac{p(z_t | x_t^{(j)})}{\lambda^{(j)}}$$

The pseudo-code algorithm for an auxiliary particle filter is given in Table 2.

Basically the APF performs a “dry run” with the current data to see which of the previous states are most promising. This is done by sampling a set of indices according to a so-called first stage weights distribution $\lambda^{(i)}$. The resampling returns a set of j_1, \dots, j_N indices (with repetitions). If the value of the index j refers to the index i of the particle i before resampling, we denote this with j^i .

During this resampling operation the promising states are reselected many times and the unlikely ones are discarded. Moreover, the APF performs resampling before state prediction and weight update, as opposed to the generic particle filter, which resamples after these operation.

Algorithm 2 Auxiliary particle filter

```

{ $x_t^{(i)}, w_t^{(i)}$ } $_{i=1}^N = APF[\{x_{t-1}^{(i)}, w_{t-1}^{(i)}\}_{i=1}^N, z_t, u_t]$ 
FOR  $i = 1 : N$ 

    evolve samples using raw odometry
 $\mu_t^{(i)} = f(x_{t-1}^{(i)}, u_t)$ 
    calculate weights using observations
 $\lambda_t^{(i)} = p(z_t | \mu_t^{(i)}) w_{t-1}^{(i)}$ 

ENDFOR
FOR  $i = 1 : N$ 

    normalize weights of samples  $\tilde{\lambda}_t = NORMALIZE[\lambda_t^{(i)}]$ 

ENDFOR
{ $j^i, -$ } $_{j=1}^N = RESAMPLE[\{x_t^{(i)}, \tilde{\lambda}_t^{(i)}\}_{i=1}^N]$ 
FOR  $j = 1 : N$ 

    sample using motion model
 $x_t^j \sim p(x_t | x_{t-1}^j)$ 
    assign weights to samples
 $w^j = \frac{p(z_t | x_t^j)}{\tilde{\lambda}_t^j}$ 

ENDFOR

```

Table 2: Pseudo-code of a localization Auxiliary Particle Filter

The APF evolves the particles using a reduced noise motion model. This is necessary because the weight update step uses the last pdf and the current observations. Adding noise after weight update give more chance to best particles to be replicated.

3 Algorithm Implementation

In this paper, we compare two different kinds of particle filters: SIR (Sampling Importance Resampling) and APF (Auxiliary Particle Filter). Given a filtering method, there are a number of implementation issues to be considered and several parameters to be tuned.

The motion model depends on the robot kinematics and on the characteristics of the environment (i.e. the friction with the surface and the presence of collisions). The observation model depends on the characteristics of both the landmarks being observed and the sensor.

In this section we first focus our attention on a motion model suitable for the system being analyzed, and subsequently we discuss an observation model taking into account the different classes of landmarks that can be observed.

3.1 Motion Model

When the robot moves, its pose estimate should be updated according to the motion model, incorporating the relative movement u_t , estimated from the odometry. In the Sony AIBO, such a displacement can be obtained by taking into account the joints measures returned from the internal motor encoders. Inaccuracies in the model, as well as environment random phenomena (such as slipping and collisions), affect reliability and precision of measuring such a displacement.

A simple motion model can take into account such a noise by adding an amount of random Gaussian noise to odometry update:

$$x_t \sim x_{t-1} \oplus (u_t + \mathbf{e}_t)$$

where \mathbf{e}_t is the random variable representing the noise affecting the odometry measure u_t , and \oplus is the standard motion composition operator defined as in [12].

A more complex motion model can also consider noise depending on the relative motion of the robot

$$x_t \sim x_{t-1} \oplus (u_t + \alpha(u_t) \cdot \mathbf{e}_t)$$

where $\alpha(u_t)$ is a matrix of functions of the odometry measurement, and \mathbf{e}_t represents the noise affecting the movement for each time a unity distance is traveled. If α is a constant, the variance of the odometry error grows linearly with the distance traveled by the robot.

Finally, we can extend the previous model considering the presence of random collisions. When a robot hits an object or another robot, it is likely to stack, although the odometry measures a non zero displacement. We can take into account this phenomenon by including a prior of hitting an obstacle and marginalizing it out, as follows.

$$x_t \sim \mathbf{h} \cdot x_{t-1} + \neg \mathbf{h} \cdot (x_{t-1} \oplus (u_t + \alpha(u_t) \cdot \mathbf{e}_t))$$

where \mathbf{h} is a binary random variable that takes into account the probability of hitting an obstacle.

3.2 Observation Model

The environment of RoboCup Four-Legged League includes a set of features that are normally used in localization: unique colored beacons (or markers), unique colored goals, and white lines on the green carpet.

Since the beacons are of limited size they are usually entirely contained in an image. The vision system on the robot can estimate the position of these beacons in the robot reference frame. With these preconditions, developing a localization algorithm should be straightforward, however the noise affecting both the robot sensing and the robot motion, as well as the dynamic environment turns this task into a challenging one.

In particular, the low resolution of the camera, combined with motion blurring effects caused by the robot movements, affects the reliability and precision of feature detection. This is particularly evident when a landmark located quite far away from the robot is perceived. In this case a beacon occupy only a few pixels in the image (as few as 10), and the estimation process is likely to fail.

An adequate observation model for a generic landmarks takes into account these phenomena is

$$p(z|x) = p(z_e) + \sum_i p(z_i|x)$$

here, $p(z|x)$ is the probability of the reading z given the robot pose x . It can be expressed as the conjunction of the following disjoint events:

- the reading is generated by a spurious reading with probability $p(z_e)$ or,
- it is due the landmark γ_i , with probability $p(z_i|x)$.

More in detail, $p(z_i|x)$ can be expressed as

$$p(z_i|x) = p(z|\gamma_i, x)p(\gamma_i|x)$$

here, $p(z|\gamma_i, x)$ is the probability of making the observation z given that it originates from landmark γ_i and the robot is in x , and $p(\gamma_i|x)$ is the prior of perceiving the landmark γ_i from the location x .

The measurement probability is a function of the angular (δ_α) and linear (δ_ρ) distance between the expected and the measured landmark locations. For every type of landmark we can use this equation:

$$\begin{aligned} p(z|b_i, x) &\propto e^{(\delta_\alpha^2/\sigma_\alpha^2)} e^{(\delta_\rho^2/\sigma_\rho^2)} \\ p(z|p_i, x) &\propto e^{(\delta_\alpha^2/\sigma_\alpha^2)} \\ p(z|l_i, x) &\propto e^{(\delta_\theta^2/\sigma_\theta^2)} e^{(\delta_\rho^2/\sigma_\rho^2)}. \end{aligned}$$

In the previous equations we use α denote angular measurements, and ρ indicates the distance. For lines across the fields (and corners, that are intersection of lines), we assume that they are expressed in polar coordinates, so ρ and θ indicates the hough parameters of a single detected line.

For single beacon observation, b_i , we use both angle and distance, while for single goal post , p_i we only use the measured angle. This is because goal posts aren't recognized exactly and they are less reliable. Each σ inside equation is specific to type of observation and are not the same.

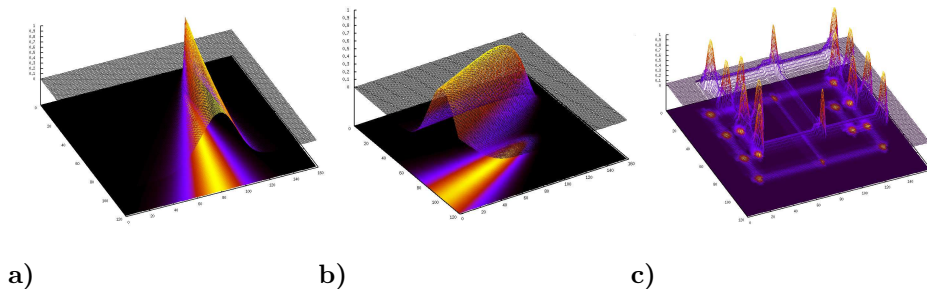


Figure 1: Graphical representations of likelihood for a) beacon, b) goal post and c) intersection of lines.

The observation models presented above are graphically shown in Figure 1. Each of them show the probability distribution function when robot sees respectively two beacons, a goal or an intersection lines.

The probability for the single observation models are then combined to form definitive probabilities as

$$p_{particles} = \prod p(z|b_i, x) \prod p(z|p_i, x) \prod p(z|l_i, x)$$

where p_{beacon} is single probability of beacons observation, p_{goal} is the same as above but referred to goals and p_{line} is referred to lines.

3.3 Sensor Resetting

During a game, the robots interact each other: they compete to control the ball, they can be obstructed during their motion, they can be penalized by the referee, etc. In such situations they have very poor observations and low reliability on odometry data. These situations makes it likely a direct implementation of MCL to fail.

Additionally, it can happen that the robot is moved by the referee in a different field location, and the robot has to re-localize itself from scratch, starting from a wrong pose estimate.

In literature many solutions have been presented, the most popular is the “sensor resetting technique”. This method breaks the Markov assumption of particle filter, and hacks the particle distribution changing it. Some authors motivated this “random injection” of particles as an attempt to model the non zero probability that the robot is kidnapped lost.

The general sensor resetting procedure acts by drawing samples from the observation density function $p(x_t|z_t)$, and adding them to the current samples. In this way, if the new particles are reliable, in the next time slices they are replicated against the bad ones.

The problem using this approach is that such replacement must be done carefully, in order to avoid replacing good samples with incorrect ones. In fact, in presence of outliers, sensor resetting may insert bad hypotheses compromising the correctness of distribution. It can be observed that if bad hypotheses are inserted by the sensor resetting routine, in the next time slices the filter should remove them. However, but in case of persistent outliers the filter is likely to converge to the wrong hypothesis.

A crucial issue in performing sensor resetting is determining the number of new particles to introduce. To this end an extension to the traditional MCL algorithm, called Adaptive MCL (AMCL) has been proposed by Fox [14]. The idea is to relate the number of particle to be introduced by sensor resetting to the observation likelihood. In order to avoid a spurious observation to cause a filter reset, the average observation likelihood is tracked over time by a pair of exponential filters. One of these filters has a slow decay constant while the other has a fast one. At each time a ratio w_{fast}/w_{slow} between the exponentially filtered observation likelihoods can be obtained by the two filters.

At each filter update a particle is replaced with probability $1 - w_{\text{fast}}/w_{\text{slow}}$. In practice, if the short term likelihood is better or equal than the long term one, no samples are added, while if it is worse samples are added according to the ratio between the two. The exponential filter provides a certain degree of rejection to temporary outliers.

The sensor resetting algorithm we have implemented for the RoboCup Four-legged scenario is based on a temporal hysteresis. It uses the best particles, chosen by its likelihood, to compute a confidence value on observations (using an history of single observations likelihood). Then this value is compared with a pair of threshold to give an idea if the robot is or not localized: after a fixed run of cycles in not localized condition, we fire the use of sensor resetting. In this way, if there are misleading perceptions, the robot do not tele-port itself to another position (the drawback is that the robot use more time to recover from bad position). In the next section we show experimental results for this method.

4 Experiments

In this section we present the results of localization experiments for the methods described in the previous section.

The experiments have been performed using standard Four-Legged soccer field as used in RoboCup 2005 and RoboCup 2006. That is different from RoboCup 2004 (with larger field and less beacon), where experiment in [6] have been performed. We also use two different strategies to track real robot position: external camera to track real position of robot and ground truth made using some measured spot and a smoothing technique.

We work in this way:

- we mark every feature on all image taken from robot;
- we use this true perception to generate a log with perfect distance and angle for object (these represent the real ground truth of perception);
- we iterate the localization task several times on same input: first iteration the localization is performed in normal way. The second iteration starts using the last set of particles and use log backward from last to first frame. This way the robot first go forward and then backward. Running localization many times we obtain a smoothed ground truth.

In this way we are able to run localization methods (possibly with different parameters) on the same input (i.e., the image sequences) several times and to compare different methods on the same input. Moreover, by using the ground-truth of robot path, we can measure absolute errors in localization.

For the experiments, we use three different navigation settings: random walk, with and without kidnapping and a playing mode.

The subsequent path graphs use these conventions: green is used for field lines, cyan for ground truth, red for SIR and blue for APF.

Random walk without kidnapping In the first experiment (the path is shown in fig 2) we consider the behavior of SIR and APF using 100 particles, when the robot is in a random walk, without obstacles.

In Fig. 3 the experiment has been repeated using the sensor resetting technique. The SIR particle filter now converges faster and recover more precisely the position. In particular, in the path from (a) to (b) there is a very low perception rate, and with the sensor resetting enabled it gives more chance to recover from wrong position. Using sensor resetting shows another fact: the APF is now less robust than previous case and more spikes are present in the calculated path.

Random walk with kidnapping The second experiment considers another random walk but with a teleporting of robot to simulate the kidnapping problem. The kidnapping was performed from (a) to (b), shown in Fig. 4.

We can see the performance of SIR using sensor resetting: when the kidnapping happen it recovers from erroneous position quickly. The APF instead changes more slowly. This behavior is a result of the algorithms which try to update using prior: when we force to use sensor resetting, some particles with inconsistent prior have chance to be resampled, especially if there are some burst of inconsistent data.

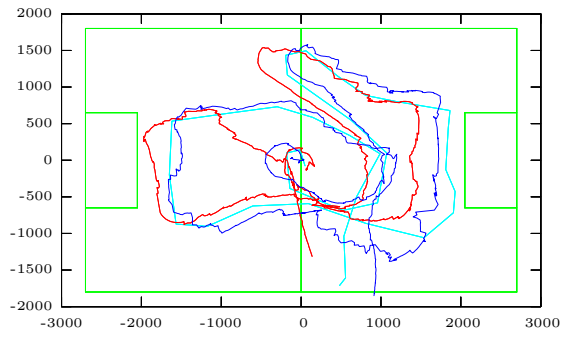


Figure 2: First experiment: random walk sensor resetting disabled.

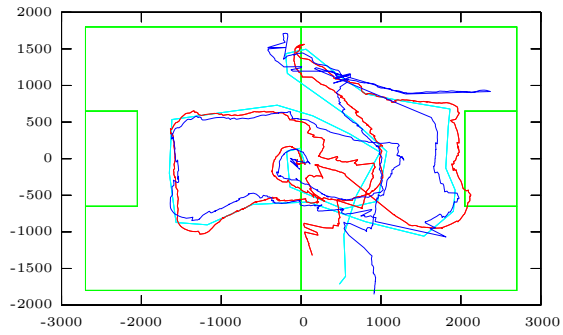


Figure 3: First experiment: random walk, sensor resetting enabled.

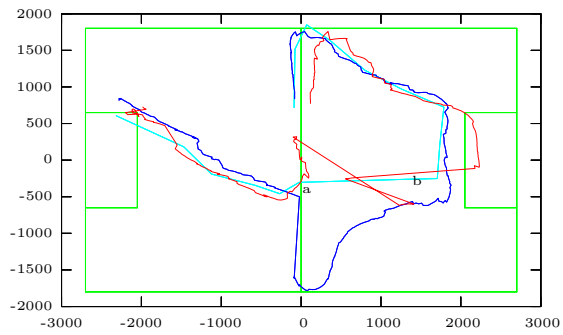


Figure 4: Second experiment: random walk with kidnapping from a) to b).

Playing mode The last experiment is done logging the robot when it is in a classical playing sequence. When robot is in playing state, it interacts with other robot, competing for ball, occasionally kicking the ball itself. In such situations the odometry give very noisy information to robot.

Following [6] we performed several experiments varying observation frequency and additional gaussian noise (see fig. 5). We enabled sensor resetting to show its impact in performance.

Sparse data We create datafile using only a fraction of observations, down to 1/256th of them. Results are shown in fig. 5:a.

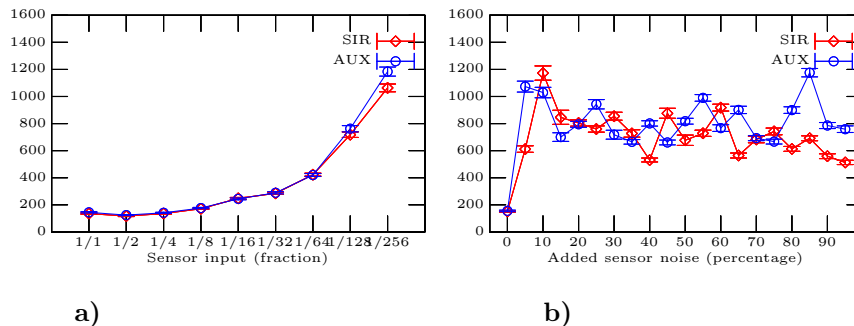


Figure 5: Performance of SIR and AUX compared a) when using only a fraction of observations and b) when additional gaussian noise is present. On the y axis is plotted the localization error in millimeters.

The difference between two methods are negligible and both degrades its performance in a similar way as in [6].

Gaussian Noise We add artificial noise depending on measurements from marker. The linear noise is approximately 15% of the real distance and the angular noise is fixed to 10. Results are shown in fig 5:b.

5 Conclusion

In this paper we have illustrated the performance of SIR-based particle filter and Auxiliary particle filter, to accomplish localization task. We shown that APF works better than SIR when not using sensor resetting, and that this method is more reliable in position tracking.

Moreover, we shown that sensor resetting technique should be used with caution, although when robot is in global positioning and it uses small set of particles this method gives some hint to rapidly converge the pose estimation.

From a more deep analysis of these behaviors it is possible to identify well defined situations in which a strategy performs better than another, given the environment and the task the robot is executing. This fact can be exploited by integrating in the localization technique information about the game state (or in general about the task state) aiming at choosing the localization strategy and setting that is more suitable for the current situation. In other words, from our experiments it appears that an appropriate selection of localization settings with respect to current state of the robot task would increase overall performance of the localization process. As future work we are investigating the effectiveness of such an integrated approach.

References

- [1] W. Burgard, D. Fox, D. Hennig, and T. Schmidt. Estimating the absolute position of a mobile robot using position probabilities grid. In *Proc. of 14th National Conference on Artificial Intelligence (AAAI'96)*, pages 896–901, 1996.
- [2] W. Burgard, D. Fox, and S Thrun. Robust monte carlo localization fot mobile robots. *Journal of Artificial Intelligence*, 2001.

- [3] A. Doucet, N. De Freitas, and N. Gordon. *Sequential Monte Carlo Methods in Practice*. Springer Verlag, 2001.
- [4] D. Fox, W. Burgard, F. Dellaert, and S. Thrun. Monte carlo localization: Efficient position estimation for mobile robots. In *Proc. of the 16th National Conference on Artificial Intelligence (AAAI99)*, 1999.
- [5] N. Gordon, D. Salmond, and C Ewing. A novel approach to nonlinear nongaussian bayesian estimation. In *In Proceedings F.*, pages 107–113, 1993.
- [6] J. S. Gutmann and D. Fox. An experimental comparison of localization methods continued. In *In Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2002.
- [7] P Jensfelt and S Kristensen. Active global localization for a mobile robot using multiple hypothesis tracking. *IEEE Transactions on Robotics and Automation*, 17(5):748–760, October 2001.
- [8] P Jensfelt and S Kristensen. An experimental comparison of localization method, the mhl sessions. In *IROS*, 2003.
- [9] Patric Jensfelt, David Austin, Olle Wijk, and Magnus Andersson. Feature based condensation for mobile robot localization. In *IEEE Intl. Conf. on Robotics and Automation*, 2000.
- [10] S. Lense and M. Veloso. Sensor resetting localization for poorly modelled mobile robots. In *Proceedings of International Conference on Robotics and Automation (ICRA '00)*, 2000.
- [11] J.J Leonard and H.F. Durrant-White. Mobile robot localization by tracking geometric beacons. *IEEE Transactions on Robotics and Automation*, 7:376–382, 1991.
- [12] F. Lu and E. Milius. Robot pose estimation in unknown environments by matching 2D range scans. *Journal of Intelligent and Robotic Systems*, 18:249–275, 1997.
- [13] M. Pitt and N. Shephard. Filtering via simulation: Auxiliary particle filters. *Journal of the American Statistical Association*, 94(446):590–599, 1999.
- [14] S. Thrun, W. Burgard, and D. Fox. *Probabilistic Robotics*. MIT press, 2005.
- [15] N. Vlassis, B. Terwijn, and B. Kröse. Auxiliary particle filter robot localization from high-dimensional sensor observations. In *Proc. IEEE Int. Conf. on Robotics and Automation (ICRA)*, 2002.