KFUPM
COE 584 – Robotics
Course Project

Robot Self- Localization in a Known Environment

RoboCup as Example

For

Prof. Mayez Al-Mohammed

By

Ahmad Salam AlRefai, Computer Engineering dept.

Mohammad Shahab, Systems Engineering dept.

June 2008

Table of Contents

1. Introduction

Here in this report, the Robot Self-Localization problem is investigated. This report will discuss several techniques in the literature to solve the localization problem. With the area of Autonomous Robotics is rapidly growing, the need of more efficient motion planning is increasing. This can not be established unless a robot can knows its own location in the environment. For a human, localization is easy with high processing quality of the visual information from the eyes. However, robots need more power to localize in the most accurate way.

Here, we discuss how a robot can estimate its location automatically in known environment. Specifically, the standard example investigated here is the RoboCup competition environment. The report will begin by giving some definition about the localization problem. Brief familiarization of the RoboCup competition is also presented. Brief literature survey about the research aspects related is also included. Then, the report will go into establishing the probabilistic framework of the problem. Then the major parts of the report will discuss in detail the Kalman Filter and the Particle Filter with orientation towards the robocup application. Further extensions to different techniques are also discussed later on. Simulation examples are also presented. The report will conclude with giving discussion about future directions in the research of robot localization problem.

2. Problem Statement

- **Autonomous Robotics**

The idea of Autonomous Robotics started in 1920 when Karel Capek introduced, in his play "Rossum's Universal Robot's", some robots that control the environment and behave independently.

Autonomy of a robot is a characteristic define how much the robots depends on prior knowledge of the environment and control from outside. Robots can be divided into three types depending on their autonomy. First, non-autonomous robots are those that completely driven by humans.Second, semi-autonomous are either controlled by human or behave independently. Fully-autonomous robot are completely driven by the robots themselves, they don't require any guidance or control in order to do their tasks.

- **What is Localization?**

Localization is actually answering the question "Where am I?" We want to know in an environment what is the pose (x & y) of the robot and what is the orientation (heading).

- Position tracking: the robot knows the initial location; the goal is to keep track of position while the robot is navigating through the environment.
- Wake up robot (global localization) robot does not know initial position.
- Kidnapped robot: the robot does exactly know where it is when it is localized but all of a sudden it is transferred or 'kidnapped to another location.

- **Inputs**

The inputs to the problem of the localization are sensor measurements or feature vector of an image. In robocup the camera is the main sensor which drives for you a lot of information about your location and also other information. We can divide the measurements into two types: Relative Position Measurements and Absolute Position Measurements. Odometry is considered as relative measurement, so you update the location according to how much you moved given that you know your previous location. The image or vision information like angle of the beacon in robocup example is considered as absolute measurement of the robot, because it is independent of how much the robot is moved or the belief about previous location. It is only dependent on the real location of the robot.

- **Outputs**

The output of localization is a belief about the robot location and orientation given to the next iteration. And also given some algorithm we feed the behavior subsystem with estimate about the location and orientation of the robot, the behavior system then decide what the robot is going to do. The location at time k can be represented as

$$x_k = \begin{bmatrix} x_{[x],k} \\ x_{[y],k} \\ x_{[\emptyset],k} \end{bmatrix}$$

- **What is RoboCup?**

Here in this section, an overview of the RoboCup competition is presented. This overview is aiming to give an idea about the environment where the localization takes place, namely, the RoboCup soccer field. To just

have an idea about the competition, this section will shed the light, in brief manner, upon the history and nature of contests.

RoboCup started in 1997 with contests of Small-size and middle-size leagues. This first RoboCup competition was held in Nagoya, Japan with 38 teams from 11 countries. This Competition grew up with the years to have 321 participating teams from 39 countries in 2007 in Atlanta, USA. The next RoboCup competition of 2008 will be held in Suzhou, China.

RoboCup is an international research and education initiative. It is to enhance Artificial Intelligence and intelligent robotics research through an interesting problem of Soccer where broad range of technologies is integrated and tested, and towards giving project-oriented education.

For a team of robots achieving the overall objectives, many aspects are investigated:

1) design principles of autonomous agents,
2) multi-agent collaboration,
3) strategy acquisition,
4) real-time reasoning,
5) robotics, and
6) Sensor-fusion.

RoboCup is a task for a team of multiple mobile robots under a *dynamic* environment.

As of 2007, in the last competition held in Atlanta, participating teams fought together in different leagues (i.e. different kinds of contests:

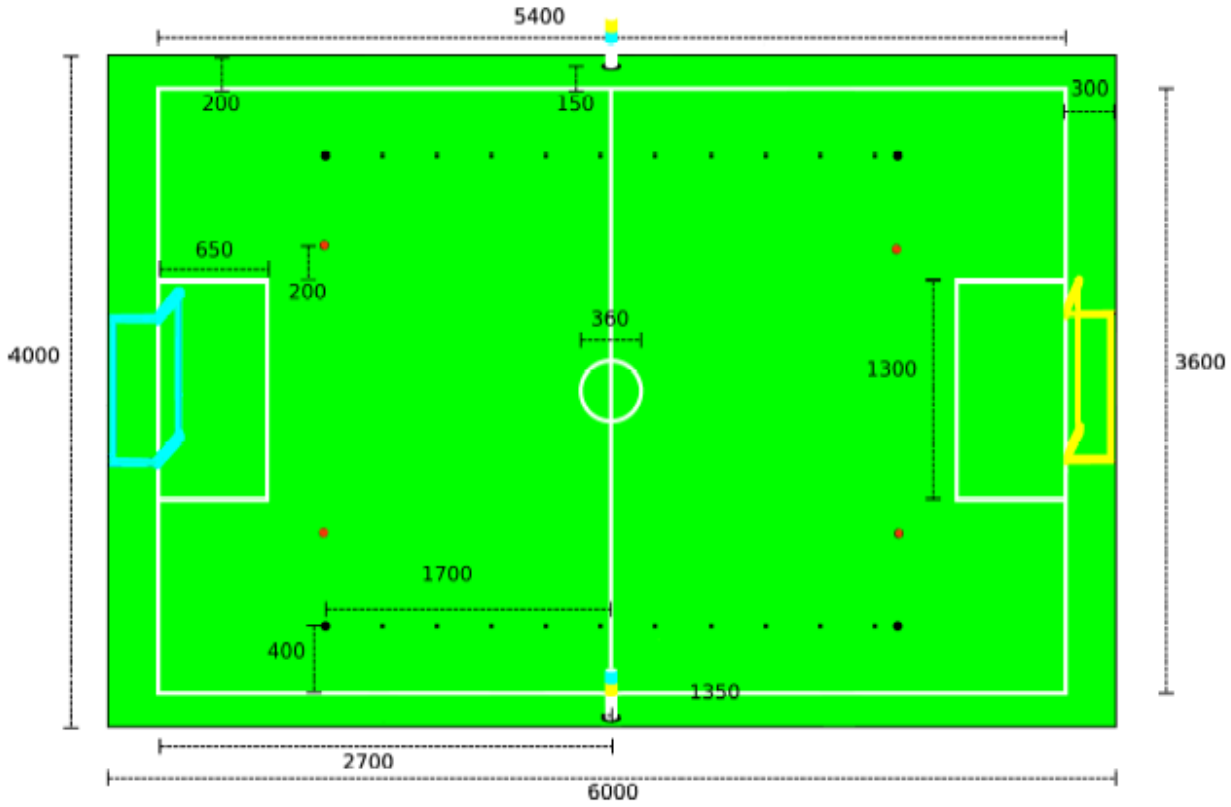
1) Simulation League
    a. 2D, 3D, 3D Development, Mixed Reality

2) Small Size Robot League
3) Middle Size Robot League
4) Four-Legged Robot League (until 2007). From 2008 it will be done in a Standard Platform.
5) Humanoid League (from 2002)
    a. Kid-size, Teen-size

In RoboCup, the environment is known; which is the soccer field. In order for a robot in some team proceed with the strategy to win the match, the ability of the robot to localize itself in the field is crucial. Further planning and strategies in the game depend heavily on the location of the robot. So, that's why in our project we are emphasizing on the area of application related, but not restricted, to soccer robot self-localization.

Soccer Field Description

In order to have the full utilization of information available for a robot, known features of the environment should be used in the localization algorithm. So, here in this section, a description of the soccer field will be given for the purpose of further analysis of the localization algorithm.



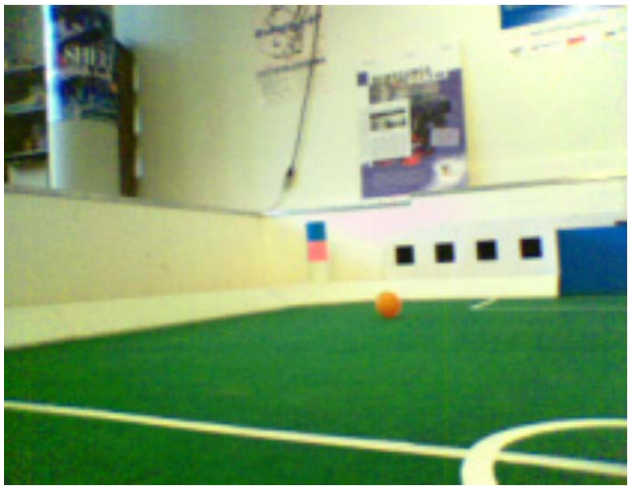
The description here will give the details for the field construction for the Four-Legged and Humanoid leagues. We are giving the description for both because of the need of self-localization in both leagues. The above figure gives the dimensions for the Four-Legged league. In order to utilize all information available, the most important 'features' needed by the localization algorithm are:

- The cylindrical **Beacon** landmarks,
- the **Goals,**
- And the field lines.

In the Four-Legged league, the beacons are put half-way in the field as shown in above figure, with the bottom section in white and other two upper sections colored in the two goal colors (blue & yellow), with different order in both beacons.

In the humanoid league, there are four cylindrical beacons at each corner of the field. The upper and bottom sections of the beacon are colored with the same color of the neighboring goal (i.e. blue or yellow). The middle section is colored with opposite goal. Field lines are white and organized as depicted in the above figure.

Another feature that could be utilized is the ball. The ball is an always-moving object colored in orange. The outside area of the field can include many other features (e.g. people) which could be unknown.



3. Literature Survey

Here, the section will give some idea about the current research on self-localization. Generally, first self-localization techniques used the concept of Triangulation to find the robots locations. But usually this method lacks accuracy because of not giving attention to noise. Some improvement to this method is done by averaging several computations.

As robotic research increased by 1990's, the need of self-localization increased also. As by 1995, different discrete approaches were used. Grid-based and metric representation of the environment was used to enhance the localization. Generally, these methods require high amount of memory which makes it not favored to fast robotic applications. Also, topological representations of the environment were used.

The localization problem is an estimation problem, with robot's position being the unknown state to be estimated. So, the work of R.E. Kalman with his Kalman Filter was the biggest discovery to be applied in general estimation problem and, in our case, robot localization. A standalone section of the report will be reserved for the Kalman Filter. Generally Kalman Filter is used for the position tracking problem. Further extensions to this filter are also discussed.

As robotic systems get more complex and environment get more dynamic, the need of more efficient techniques are demanded. The research work for further complex robot localization problems were discussed by (Fox, Burgard, Dellaert & Thrun 1999), namely the Monte Carlo Localization (MCL). The MCL technique started in the 1970's and further years were used in computer vision. This method is coming from the field of Particle Filters which was been utilized mainly in robot localization problem. Particle Filters are often associated with the problem of global localization. Particle Filter (and of course Monte Carlo Localization) will be discussed with more detail in other section of the report.

Further extensions to different techniques were also attempted with testing Multi-Hypothesis problems utilizing multiple kalman filters. Other mixed techniques of Particle & Kalman Filters were investigated also.

4. Assumptions & Models

Assumptions

The environment is assumed to be known, and the environment should at least be partially observable.

Motion Model

The motion model describes how we can update the location information according to the previous location of the robot. The new pose of the robot is calculated by summing up the previous pose and how much he moved with some noise. This is done according to the following equation.

$$Pose_{new} = Pose_{old} + \Delta_{\text{odometry}} + \Delta_{\text{error}} \quad (1)$$

Observation Model

The observation model links the observation of the robot to the location of the robot. This model calculate what the robot should observe if it was at specific locations. In robocup the angles to the vertical lines of the beacons are used as perceptual landmark in updating according to the motion model.

Noise Models

According to your environment you can model your noise, so some teams they use the knowledge of the robot and their environment and according to some study they update the compensate the errors that are in the noise model.

5. Probabilistic Framework

Here in this section, the report will start giving the solid mathematical foundation in order to present the localization techniques. The nature of the localization problem is probabilistic. This is due to the noisy dynamic changing environment. We can see the problem probabilistically as that the robot has a ***belief*** about its location. So, at one time instant, we do not depend on one possible location, but we can consider the whole space of locations. With good utilization of all available information to the robot, one can be certain about one location with some confidence. So, the localization problem can be summed up to the process of estimating the Probability Density of the whole location space (e.g. the soccer field).

The advantages of using the notion of randomness (probabilistic) can include that it:

- Can accommodate inaccurate models
- Can accommodate imperfect sensors
- Is Robust in real-world applications
- Best known approach to many hard robotics problems

However, in the same time, disadvantages can emerge from this framework as it is:

- Computationally demanding
- Has False assumptions
- Approximate!

- **Bayes Filter**

The localization problem can be put probabilistically as a Bayesian Estimation Problem, which is related to the well-known Bayes Rule of

$$P(x|y) = \frac{P(y|x) \cdot P(x)}{P(y)} = \frac{\text{Likelihood} \cdot \text{Prior}}{\text{Evidence}}$$

The Bayesian framework is developed with the work of Markov Localization. The Markov Localization utilizes the available information from multiple sensors in the form of the relative and absolute measurements to give an estimation of probability density of the belief.

- **Belief**

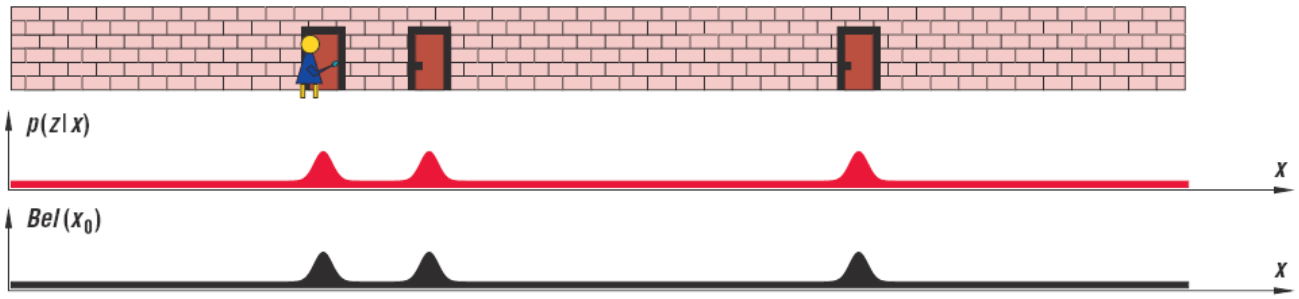
A robot has a belief about its location. This belief represents the probability of the location (or *pose*) $x \in \mathbb{R}^m$, with m being the number of states defining the location of the robot. Here in a usual applications, m = 3 with

$$x = \begin{bmatrix} X \\ Y \\ \theta \end{bmatrix}$$

With X being the x-coordinate, Y being the y-coordinate and θ being the robot heading. Here, as it is obvious, we assume that the robot moves only in a 2D plan. Further investigations can be done for 3D motion, as for Unmanned Aerial Vehicles (UAVs). So the belief of the location of the robot can be represented as

$$Bel(x_k) = P(x_k|d_0, d_1, d_2, \dots, d_k)$$

So, the probability density of the location of the robot at time *k* can be evaluated as the probability of the robot being in certain location x_k given all available information from the beginning of the process (d_k being all available information at time *k*). The available information could of course include the environment map or any other a-priori knowledge of the process. Higher probability for a certain location means high possibility for the robot to be at that location. So, the goal of the localization is to approximate the real robot location belief. The real belief has one high peak at the real location with zeros otherwise. So, the localization tries to achieve the same belief of the real location. You can see below a picture about the belief at some time.



The available information to the robot comes into two categories: Actions & Sensing. For a robot, an action is represented by commands sent to the actuators of robot, e.g. motors. Sensing information come from the sensor-based measurements available to the robot.

So, in order to incorporate the action and sensing data in our belief, we must have some probabilistic representation of this available information. For some time *k,* a robot accepts some **actions** that change its state. We can represent this by

$$P(x_k|x_{k-1}, a_{k-1})$$

With a_k being the action sent to the robot as time k. For the sensory information, we can represent the probability of the sensor data for certain location by

$$P(s_k|x_k)$$

With s_k being the sensor data available at time k. However, the raw sensor data are of high dimension (e.g. image frame). So, to utilize the information in some efficient way, features are extracted from the sensor data giving us some feature vector z_k. So for example, s_k could be the image while z_k could be representing the landmark metric information.

- **Recursive Bayesian Equation**

In general, the localization technique aims to find and improve the belief by good utilization of the action and sensing data. So one belief can be viewed as

$$Bel(x_k) = P(x_k|z_0, a_0, z_1, a_1, \dots, z_{k-1}, a_{k-1}, z_k)$$

The above formula gives us more elegant picture of how the belief evolves. With applying the Bayes rule, the belief transforms into

$$Bel(x_k) = \frac{P(z_k|x_k, z_0, a_0, z_1, a_1, \dots, z_{k-1}, a_{k-1})P(x_k|z_0, a_0, z_1, a_1, \dots, z_{k-1}, a_{k-1})}{P(z_k|z_0, a_0, z_1, a_1, \dots, z_{k-1}, a_{k-1})}$$

But here, we assume that the process follows to the Markov Assumption. The Markov assumption which states that given knowledge of the current state, the past is independent of the future, and vice-versa. So we can have the belief as

$$Bel(x_k) = \frac{P(z_k|x_k)P(x_k|z_0, a_0, z_1, a_1, \dots, z_{k-1}, a_{k-1})}{P(z_k|z_0, a_0, z_1, a_1, \dots, z_{k-1}, a_{k-1})}$$

So, we can see that the previous state should contain all information from past. With the use of the rule of Total Probability, the rightmost term in the numerator can be changed. Also, the denominator can be assumed to be a normalization parameter because is independent of the state. So the whole belief becomes

$$Bel(x_k) = \eta_k P(z_k|x_k) \int_{\forall x} P(x_k|\, x_{k-1}, a_{k-1}) Bel(x_{k-1}) dx_{x-1}$$

With η_k being the normalization factor; and $Bel(x_{k-1})$ being the previous belief about the previous location of the robot. You can see that the above formula is a recursive equation. This equation defines the localization problem in general. The above equation sets the basic formula to 'solve' the localization problem. The evaluation of this equation has different techniques. In the following sections of the report, the major two techniques are discussed, namely, the Kalman Filter and the Particle Filter.

6. Kalman Filter Localization

- **History**

Kalman Filter is an estimator that estimate the state of a linear dynamic system, this estimator is recursive algorithm. It is named after Richard Kalman who discover this idea in 1960. It is considered as one of the greatest discoveris in the history of statistical estimation theory. It is used for state estimation in dynamic system control. It is used to control manufacturing processes like aircrafts, spacecrafts and robots.

Kalman Filter is a state estimator that woks on predication-correction algorithm. This means that the robot predict a belief about its states using the motion model and the dynamics of the system. After that it corrects its belief by the use of the measurement information.

Kalman Filter estimates the probability of being at specific location x_k given available information $z_1, \dots, z_k$. This called Belief…

$$Bel(x_k) = P(x_k|z_1, \dots, z_k)$$

By the use of Bayes' rule, total probability theorem and Markov assumption, we get

$$Bel^-(x_k) = P(x_k|z_1, \dots z_{k-1}) = \int P(x_k|x_{k-1}) Bel^+(x_{k-1}) dx_{k-1}$$

$$Bel^+(x_k) = P(x_k|z_1, \dots, z_k) = \frac{P(z_k|x_k) Bel^-(x_k)}{P(z_k|z_1, \dots, z_{k-1}}$$

The prior belief is the conditional probability of being at specific location given all the observation up to last step. The posterior Belief incorporate the last measurement into the belief. In order to calculate belief we need to know the system mode $P(x_k|x_{k-1})$ and measurement mode $P(z_k|x_k)$. The computation of the prior belief is considered as prediction of the state after a time step. The computation of the posterior belief can be seen as a correction step, which corrects the prediction step because of the noisy system.

- **Assumptions**

In order to apply Kalman Filter use both phases of prediction and correction, the Kalman filter assume that the system is linear dynamic system. That mean that we can describe the evolution of the system through the use of linear equations, and the equations that explain the relation between measurements and the system can be represented in linear equations.

The dynamic system can have some noise; however this noise must be independent of other noise sources. The noise is assumed to be white which means that the noise has noise has power all frequesnsites, and the amount of noise is not correlated in time, so knowing the amount of noise at specific time does not help in predicting the amount of noise at any other time. The noise has zero-mean implies the error is random. The noise is Gaussion, means that the distribution of the noise is modeled by bell-shaped curve. So the system and measurements noise can by described as

$$w_k \sim N(0, Q_k)$$

$$v_k \sim N(0, R_k)$$

Where $N(\mu, \Sigma)$ is a Gaussian distribution with μ mean and Σ covarience.

Gaussianity in noise implies that the distribution of the system and measurement model can be fully described by mean and variance if the motion model can be described as

$$x_k = Ax_{k-1} + w_{k-1}$$

w_{k-1} is the system noise and it is zero mean system noise with covariance Q_k, so the probabilistic system model can be

$$P(x_k|x_{k-1}) = N(Ax_{k-1}, Q_k)$$

And using the measurement model shown below

$$z_k = Hx_k + v_k$$

Similarly, by the use of gaussinity of v_k, we can write the conditional probability of seeing z given x can be written as

$$P(z_k|x_k) = N(Hx_k, R_k)$$

- **Algorithm**

The algorithm of Kalman Filter consists of an initialization step followed be an alternating prediction and corrections steps.

Initialization

Initial posterior estimate and initial uncertainty in state estimate, the distribution of the belief is $Bel^+(x_0) = N(\hat{x}_0^-, P_0^-)$. In order to initialize with initial posterior state estimate $\hat{x}_0^+$ and initial uncertainty P_0^+.

Prediction Equations

In this step the prior belief $Bel^{-}(x_k) = N(\hat{x}_k^{-}, P_k^{-})$ shall be computed where, this is done by projecting forward the most resent belief.

$$\hat{x}_k^{-} = A\hat{x}_{k-1}^{+}$$

$$P_k^{-} = AP_{k-1}^{+}A^T + Q_{k-1}$$

Where A is a matrix describes how the system evolve from the posterior of previous step to prior of the current step. Since the system is subject to noise the uncertainty P_k^{-} is increased.

Correction Equations

The correction equations are called measurement update equations. These equations correct the belief of the prediction by incorporating information from measurements. The posterior belief is computed $Bel^{+}(x_k) = N(\hat{x}_k^{+}, P_k^{+})$ according to the following equations

$$\hat{x}_k^{+} = \hat{x}_k^{-} + K_k(z_k - H\hat{x}_k^{-})$$

$$P_k^{+} = (I - K_k H)P_k^{-}$$

$$K_k = P_k^{-}H^T(HP_k^{-}H^T + R_k)^{-1}$$

In which K_k is the Kalman gain, and the posterior state $\hat{x}_k^{+}$ is calculated by adding the prior state to the difference between the measurement z_k and the predicted measurement $H\hat{x}_k^{-}$, where H is the measurement matrix given in the measurement model. The difference between the measurement z_k and the expected measurement is called the innovation. The Kalman gain give us how much of the innovation should be taken into account to determine the posterior state estimate. If the new measurement is "trusted" then the Kalman Gain will be high and the uncertainty will be close to zero and vice versa.

7. Particle Filter Localization

Here in this section, the method of particle filter will be discussed. Unlike the Kalman Filter, the particle filter is applied on problems with no assumption about the nature of noise or the nature of robotic system at hand. The particle filter has many names in the literature. Particle filter can be found named as: Bootstrap filter, Monte Carlo Localization (MCL), Survival of the fittest or Condensation algorithm.

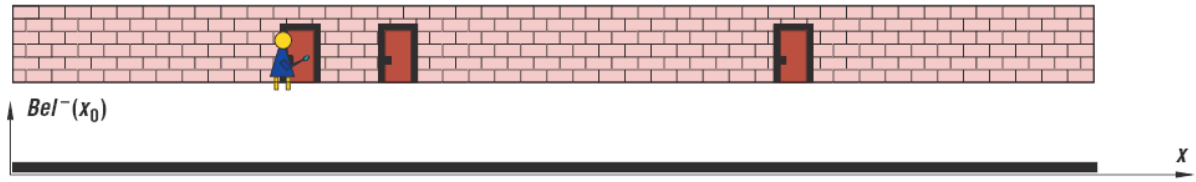
The key idea of the particle filter is to represent the previous belief about the location of the robot by a collection of weighted 'particles'. This collection of particles defines the probability density of the location. In our problem of robot localization, the particles are being some possible locations (states) of the robot. So, the belief can be represented as

$$Bel(x_k) = \{x^i, w^i\}$$

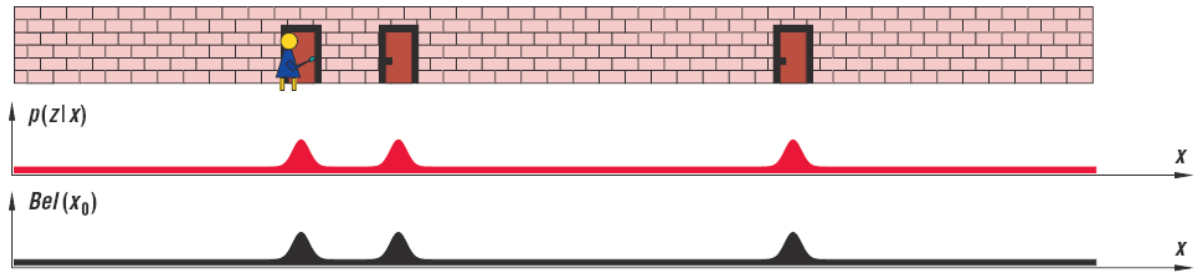
With *i=1,2,3,…, N*. with *N* being the number of particles. With w^i being the associated weight of particle *i*. Those weights are with the property of

$$\sum_{i=1}^{N} w^i = 1$$

So, for a global localization problem (i.e. robot does not know is initial location), the initial weights can be uniform as shown in the below picture.



The weights of the particles represent, in some way, the confidence about each particle (i.e. of each location). So, unlike the Kalman filter, rather than the belief being Gaussian, the belief here is represented only by a collection of discrete particles that represent the evolution of the system location. So, operating on these particles should give us a good result about the belief. You can picture a possible non-Gaussian belief by referring to the below figure.



You can see from above figure that the belief for some time instant is not following the Normal (Gaussian) distribution. The particle filter technique can be considered also as a Bayesian problem. So, recalling the recursive Bayesian equation

$$Bel(x_k) = \eta_k P(z_k|x_k) \int_{\forall x} P(x_k|\, x_{k-1}, a_{k-1}) Bel(x_{k-1}) dx_{x-1}$$

For the particles, this recursive equation can be represented as

$$Bel(x_k) = \eta_k P(z_k|x_k) \sum_{\forall N\ particles} P(x_k|x_{k-1}, a_{k-1}) Bel(x_{k-1})$$

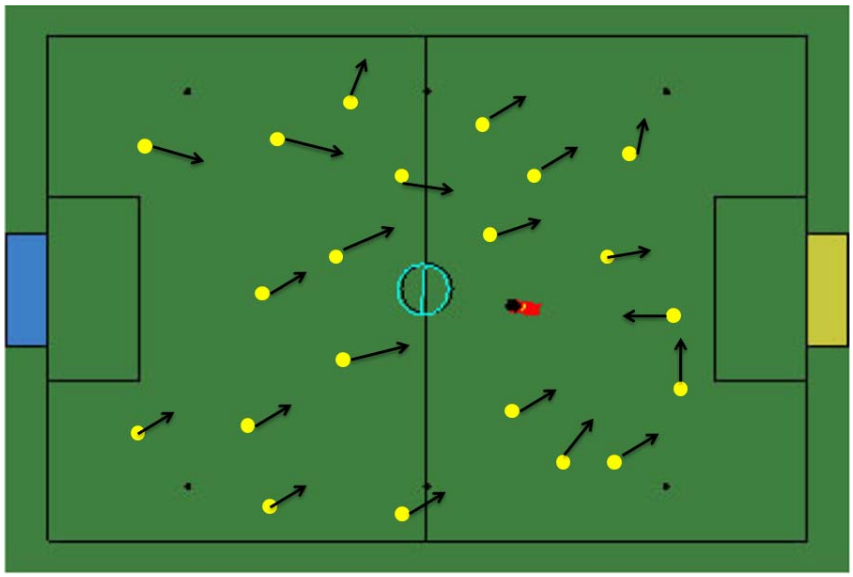
So, to utilize the above equation, the motion model and the observation models should be used in order to have better belief, i.e. better particles.

1) Motion Update

At any time instant, the algorithm generates *N* particles from the previous belief. The motion update is applied to all particles. The motion update is done using the motion model of the robot system. The motion information of the robot can be summed into the change of the odometry data of the system as

$$x_k = x_{k-1} + \Delta x$$

With Δx is the odometry update on the state of the robot. You can picture the motion update for each particle as viewed in the below figure.



In the above figure, you can see that the arrow is representing the motion update effect on each particle. The distances of the update are all the same for all particles (of course). The headings of each particle are different because of different heading values from the previous belief. This step of the algorithm (motion update) represents the evaluation of the part of the recursive equation that is

$$\sum_{\forall N\ particles} P(x_k|x_{k-1}, a_{k-1}) Bel(x_{k-1})$$

You can easily relate the motion update to the specific variables appearing in the above equation. So, the motion update propagates the previous belief to the next step, the observation update.

2) Observation Update

The evaluation of the recursive equation is left with the part of

$$\eta_k P(z_k|x_k)$$

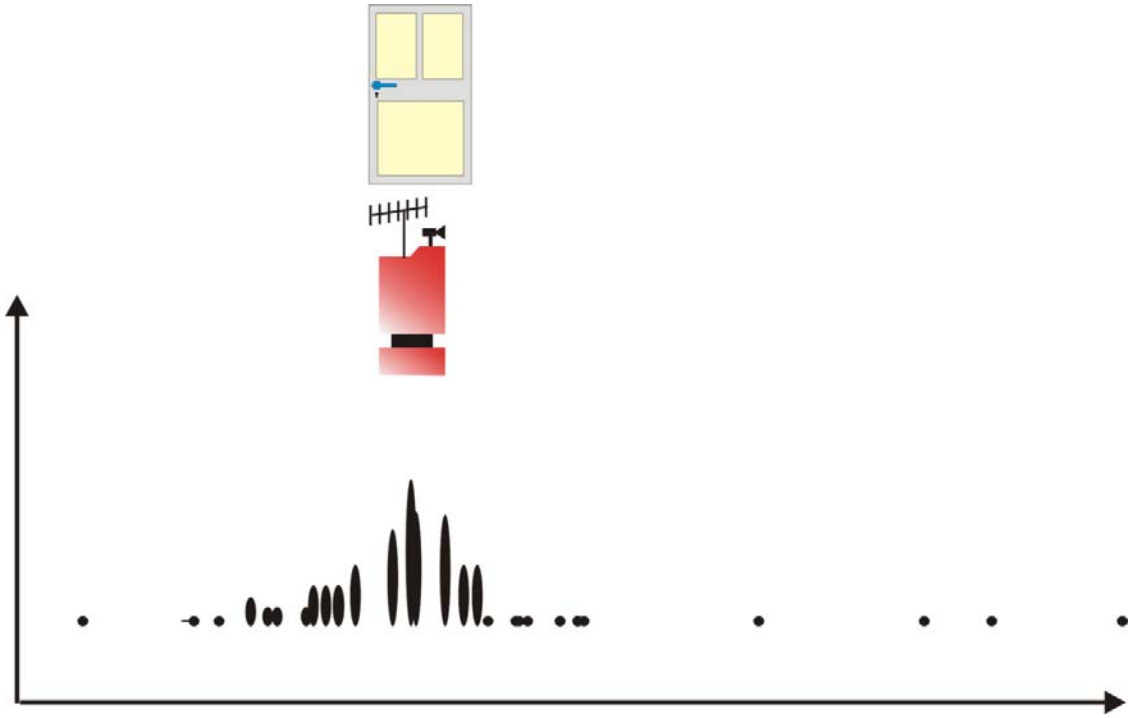
You can see easily that this part is related to the observations or sensory data available in time k. so the choice of the weights of each particle should be selected in the best way to reflect the real belief of the location of the robot. So one weight selection can be seen as

$$w^i \propto P\left(z_k\middle|x_k^i\right)$$

With x_k^i is the state of particle i. you can see here that the 'importance' weight of a particle is proportional to the effect of the sensory data on the particle state. So we can make

$$w^i = \eta^i P\left(z_k\middle|x_k^i\right)$$

with η^i is the normalization constant that makes $\sum_{i=1}^{N} w^i = 1$. So the observation update is utilizing the incoming sensory data to change the 'confidence' of each particle depending on the observation model of the system. So $P\left(z_k\middle|x_k^i\right)$ is representing the observation model of the system that evaluates the probability of available measurements given specific location of the particle. A pictorial view of the observation update can be seen in the below figure.

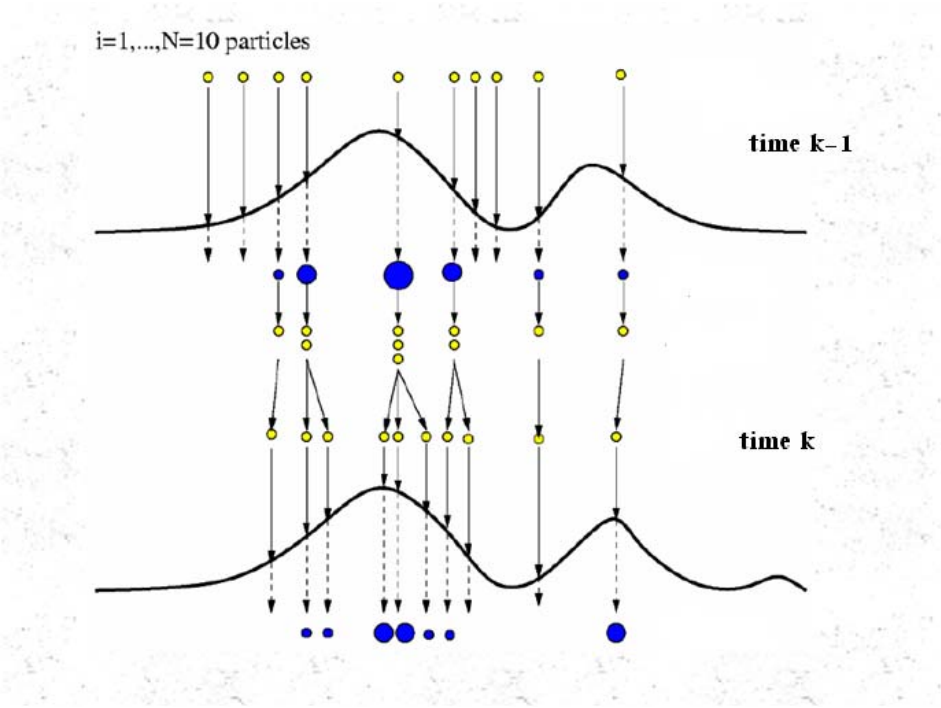


You can see that weights of particles change from one particle to another. These differences are due to the measurements coming that indicate the information of the existence of a door which increases the confidence for particular location in the space represented by the particles. To proceed to the next step, the particles should be re-generated to represent more the discrete belief of the current step.

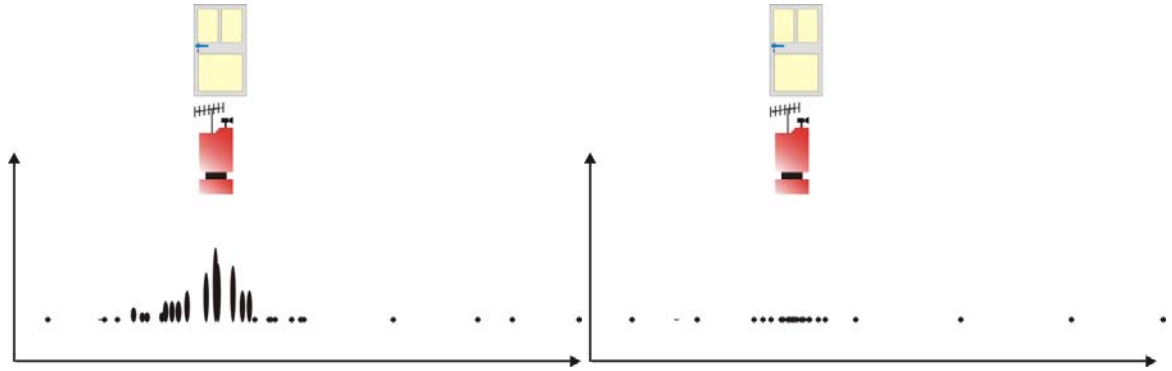
3) Resampling

In this step, the goal is to create a new set of equally weighted particles, again, by sampling the distribution of the weighted particles produced in the previous step. This should approximate the belief of the location of the robot. Different techniques are there on how to draw particles from the previous step. On method is to use the same particles from previous step, i.e. no resampling. But this will produce all but one particle to have negligible weights. This makes huge computation effort for particles whose contributions are almost zero. So the basic idea of resampling is to eliminate particles with small weights and concentrate on particles

with large weights. You can visualize a resampling for particles from time k-1 to the new step at time k in the below figure.



In the above figure, it can be seen that at time k, more particles are put around the higher-weighted particles from time k-1. This resampling can eventually eliminate the unwanted particles. But in general, resampling is also problematic. We can say that resampling limits the opportunity to parallelize since all particles need to be combined. Resampling could cause the loss of Diversity among samples, we have many repeated particles (sample impoverishment), i.e. that the new 'repeated' particles could not really represent the actual belief but rather only the previous weight. Also, In the case of very small noise, all the particles will collapse to a single point within a few iteration which produces unwanted computation.



In the above figure, on the left, the calculated weights are shown after applying the observation update. In the right, the resampling of the particles in the next step is done according to 'importance' or 'confidence' information coming from the previous weights.

4) Location (state) Estimation

In this final step for a particle filter, the location of the robot should be estimated from the current belief available. There are many ways to compute the location to be further used in future:

- One way is to calculate the mean of all particles, but this could include wrong information from wrong-weighted particles.
- Another way is to just take the particle with highest weight, but this loses the information provided by other particles.
- Another convenient way is a mixture of above ones. The location of the robot could be estimated by firstly divide the location space of the robot to many clusters according to metric description of the environment. Then, in each cluster, the mean is calculated for the particles in that cluster. Then the highest mean is accepted.

A final word about Particle filters is to mention some advantages and disadvantages. Advantages to using particle filters (or MCL):

1. Able to model non-linear system dynamics and sensor models
2. No Gaussian noise model assumptions
3. In practice, performs well in the presence of large amounts of noise and assumption violations (e.g. Markov assumption, weighting model…)
4. Simple implementation

However, Particle filter also has some disadvantages:

1. Higher computational complexity
2. Computational complexity increases exponentially compared with increases in state dimension
3. In some applications, the filter is more likely to diverge with more accurate measurements

8. Localization Method Extensions

- **Extended Kalman Filter**

The basic Kalman solves the problem of prediction in a noisy environment. But what if the motion and measurement models are described by nonlinear functions? We need to linearize the system by evaluating the Taylor series at trajectories that are updated with latest state estimates.

The Algorithm of the extended Kalman filter is ver similar to the Kalman taking into account the linearization at each step

Initialization

The system is initialized with initial posterior state estimate $\hat{x}_0^+$ and initial uncertainty P_0^+.

Prediction Equations

In this step the prior belief $Bel^-(x_k) = N(\hat{x}_k^-, P_k^-)$ shall be computed where, this is done by projecting forward the most resent belief.

$$\hat{x}_k^- = f(\hat{x}_{k-1}^+)$$

$$P_k^- = A_k P_{k-1}^+ A_k^T + Q_{k-1}$$

Where A_k is the Jacobian matrix that contains the partial derivatives of the system function $f(.)$ with respect to state x. This is evaluated at the posterior state estimate $\hat{x}_{k-1}^+$ of last time step

$$A_k = \frac{\partial f(x)}{\partial x}|_{x=\hat{x}_{k-1}^+}$$

Correction Equations

The correction equations are called measurement update equations. These equations correct the belief of the prediction by incorporating information from measurements. The posterior belief is computed $Bel^+(x_k) = N(\hat{x}_k^+, P_k^+)$ according to the following equations

$$\hat{x}_k^+ = \hat{x}_k^- + K_k(z_k - h(\hat{x}_k^-))$$

$$P_k^+ = (I - K_k H_k)P_k^-$$

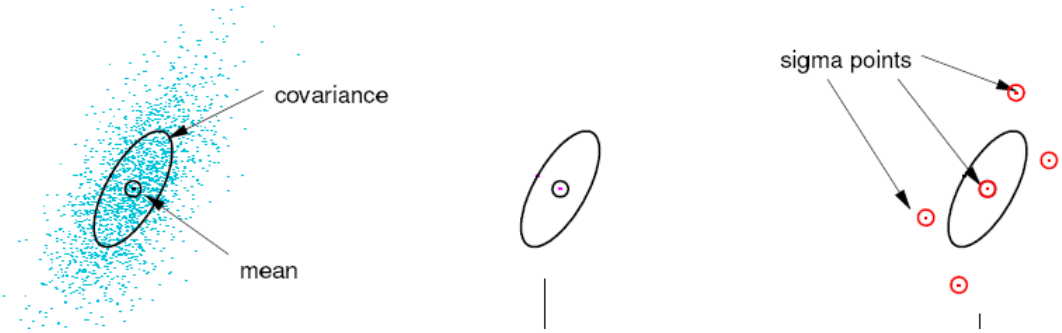
$$K_k = P_k^- H_k^T (H_k P_k^- H_k^T + R_k)^{-1}$$

Where H_k is the Jacobian Matrix that contains the partial derivatives of the measurement function $h(.)$ with respect to state x, evaluated at the prior state estimate $\hat{x}_k^-$,

$$H_k = \frac{\partial h(x)}{\partial x}|_{x=\hat{x}_k^-}$$

- **Unscented Kalman Filter**

We faced some assumptions in the normal Kalman Filter method. These assumptions could ruin the results of the filter. These assumptions that could have negative effect on the localization result are: System Linearity & Gaussian environment. But it is easily understood that the robot systems are not linear at all. Robot mechanics produces nonlinearities in the process. Extended Kalman Filter (EKF) solves that problem of nonlinearity. However, linearization error, in some application, gives false filter results at the end. So the need for another method is investigated. So, we can now present the so-called Unscented Kalman Filter (UKF).



In the above figure, the first (right) picture shows the complete distribution (or Belief) represented by particles, as in particle filter. In the middle, the kalman filter accepts only the values of the mean and covariances to describe the complete distribution (that is in the linear case or in linearized case of EKF). However, in the third (left) picture, a new concept is introduced. This is called the 'Sigma-points'. The Unscented transformation is applied to the belief of the probability density to have a collection of sigma-points which are sufficient to approximate the whole distribution. So, these Sigma-points are to approximate the belief rather than using only the mean and covariances. The number of sigma-points calculated is equal to = 2*m+1, with m being the number of states in the robot location. In our case m = 3. The selection of the sigma points is done as

$$\begin{aligned} \boldsymbol{\mathcal{X}}_0 &= \bar{\mathbf{x}} \\ \boldsymbol{\mathcal{X}}_i &= \bar{\mathbf{x}} + \left(\sqrt{(n_x+\lambda)\mathbf{P}_x}\right)_i \\ \boldsymbol{\mathcal{X}}_i &= \bar{\mathbf{x}} - \left(\sqrt{(n_x+\lambda)\mathbf{P}_x}\right)_i \end{aligned}$$

With the first sigma-point $\boldsymbol{\chi_0}$ being the mean. Excluding the first point, the above last 2 equations, each applied for different state. For example for m=3, the 2 equations are evaluated 3 times with

$$\left(\sqrt{(n_x+\lambda)\mathbf{P}_x}\right)_i$$

Being the *i*th row associated with state *i*. here, we have λ is a spread factor (to scale the spread of the sigma-points) and $\boldsymbol{P_x}$ is the covariance matrix of the states, i.e. uncertainty. Weights are chosen to normalize the points. The procedure for localization problem to be solved by UKF is to just, in every step, apply the normal Kalman Filter to each point and then normalize.

- **Unscented Particle Filter**

Here, nothing new is presented except the mixing between the particle filter technique with the concept of unscented kalman filter. The algorithm goes as:

1. At each step for each particle:
    - Calculate the Sigma Points
    - Apply Kalman Update Equations
    - Normalize and get mean and covariance for each particle

2. Continue the Particle Filter as explained before

It looks like it has more computation, but if particles number is small it will reduce computation and increase accuracy. Another variation to this is to have mixed filter. It was called 'mixed fast particle filter'. This algorithm goes as

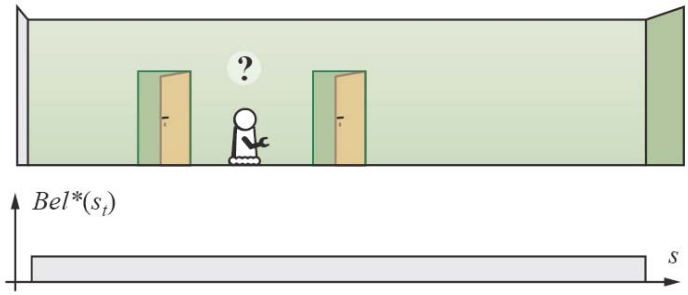
1. For p% of the N particles:
    - Apply Unscented Particle Filter

2. For (100-p)% of the particles:
    - Apply the normal particle filter
3. *Normalize* all weights from 1,2
4. Resampling

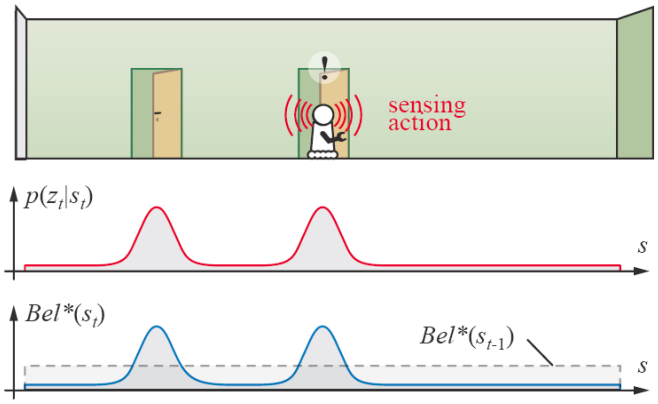
- **Negative Information**

One of the proposed techniques in localization is to make use of what you don't see, or what is call negative information. The idea of negative information is to say that if I am in this location I should get the following information or see that thing but I am not seeing it, so I know that I am not at this specific location. Negative Information gives us less information than positive information because there might be to reasons to not seeing anything either because it is really not there, or the robot could not detect its existence. Although negative information gives us less information than the positive information, it is needed especially when we do not have enough information given through the positive information. The following figures show the use of negative information.

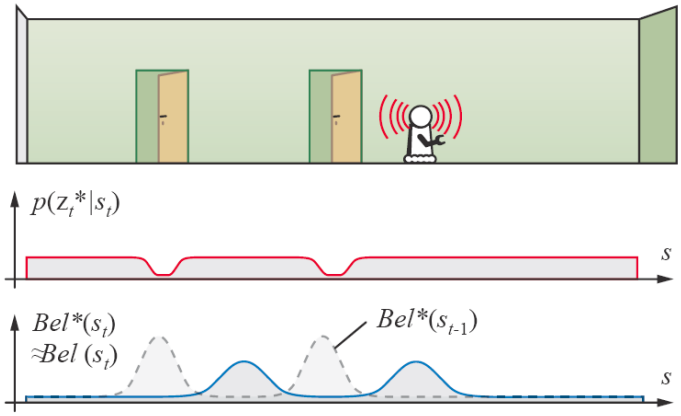
The robot initially does not know where it is located, it has a sensor to detect doors. The distribution of the belief is uniform.



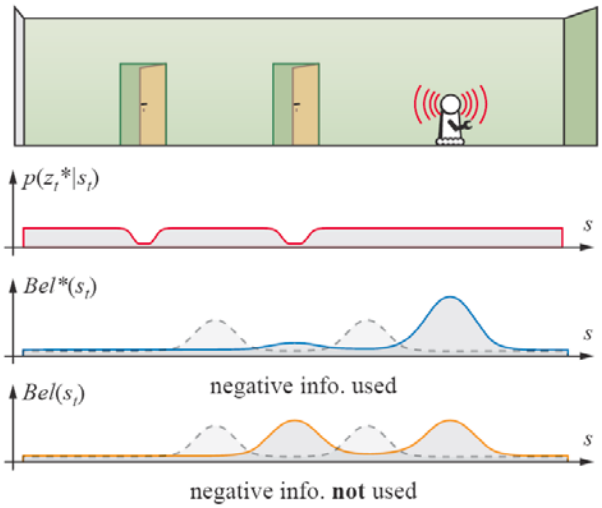
Now the robot detects a door, and since it has a prior knowledge of the environment the distribution of observing a door will increase at the location of the two doors.



The robot moves forward, it update its location according to the motion model. In this stage the negative information is negligible.



The robot moves on and he was not able to detect another door. The $Bel^*(s_t)$ shows the belief if we take the negative information into account, whereas $Bel(s_t)$ is without taking the negative information into account. The negaitive information allow the robot to remove the left peak in the distribution.



MATHEMATICAL MODELING

$$Bel^-(s_t) \leftarrow \int p(st \mid s_{t-1}, u_{t-1}) Bel(s_{t-1}) ds_{t-1}$$

$$Bel(s_t) \leftarrow \eta p(z_t \mid s_t) Bel^-(s_t)$$

$$P(z_{l,t}^* \mid s_t)$$

$$P(z_{l,t}^* \mid s_t, r_t, o_t)$$

t : Time
l: Landmark
z: Observation
u: action
s: State
*: negative information
r: sensing range
o: possible occlusion

ALGORITHM

$$Bel^-(s_t) \leftarrow \int p(st \mid s_{t-1}, u_{t-1}) Bel(s_{t-1}) ds_{t-1}$$

if (landmark l detected) **then**

$$Bel(s_t) \leftarrow \eta p(z_t \mid s_t) Bel^-(s_t)$$

else

$$Bel(s_t) \leftarrow \eta p(z_{l,t}^* \mid s_t, r_t, o_t) Bel^-(s_t)$$

end if

The algorithm for implementing the use of Negative information is in the following way

$$Bel^-(x_k) = \int p(x_k|x_{k-1}, u_{k-1}) Bel(x_{k-1}) dx_{k-1}$$

If (landmark l detected) then

$$Bel(x_k) = \eta p(z_k|x_k) Bel^-(x_k)$$

Else

$$Bel(x_k) = \eta p(z_{l,k}^*|x_k, r_k, o_k) Bel^-(x_k)$$

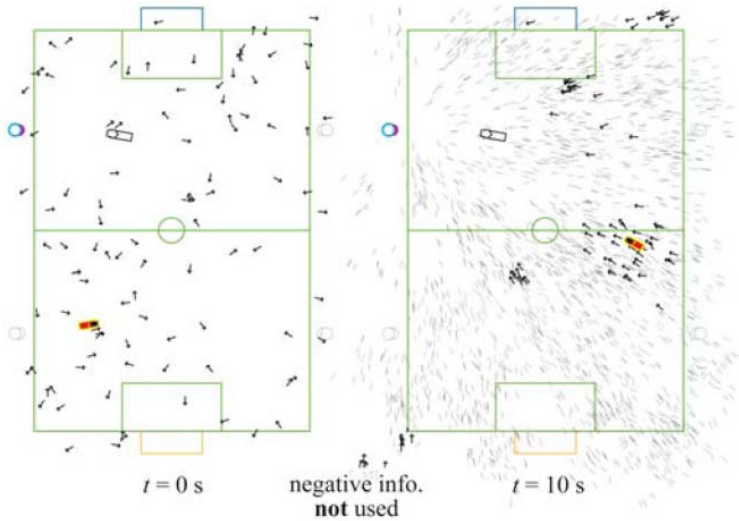
So this algorithm update the prior belief then checks if landmark is expected to be detected and it is detected it update the posterior normally according to the positive information. Otherwise it update the belief using the negative information taking into account the sensor range r, and possible occlusion o.

Using this algorithm with particle distribution in the following cases

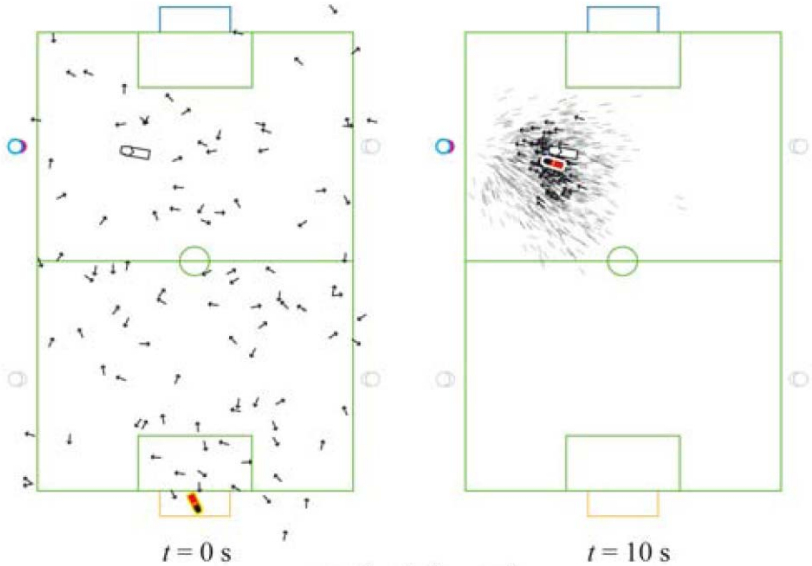
- 100 Particles (MCL) VS 2000 Particles to get better representation.
- Not using negative Information VS using negative information.

- Entropy H (information theoretical quality measure of the position estimate.

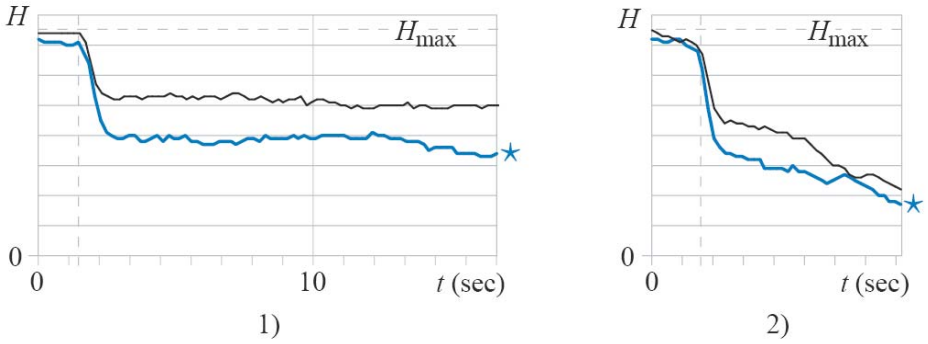
In case we don't use negative information, the robot is not able to localize in both cases whether we have 100 particle or 2000 particles, noting that only using bearing to landmark is used as a positive information.



However when we used negative information the robot is able to localize quickly as shown in the next diagram.



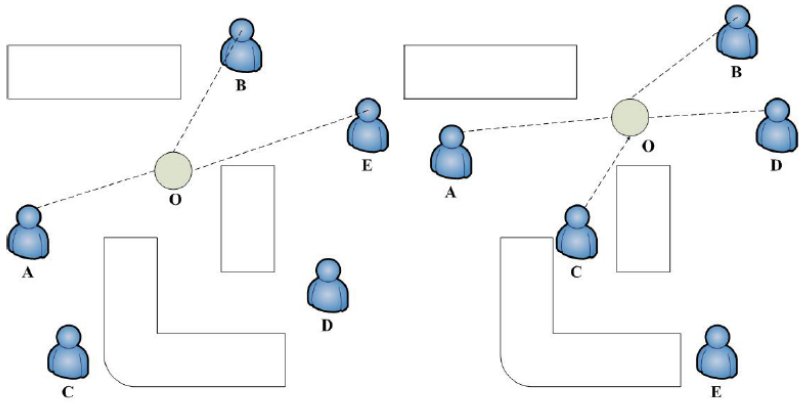
The following diagram shows the effect of using the negative information. The one to the left is with using the bearing to beacons as information, the one with * and blue is with the use of negative information. The use of negative information reduces entropy. The diagram to the right show when we use the field line as input, it improve the accuracy (reduce uncertainty) but also when we use negative information it improves it more.



- **Cooperative Localization**

Here in this section, an interesting method is discussed. This work is related to the work of 2006 team s**harPKUngfu**, of Peking University, China. They suggested the concept of "**Dynamic Object Reference**". Dynamic Object Reference can be explained through this example: A human can self-localize himself by putting, for example, special building as a reference (static). However, with Mobile Robots & Dynamic Environment, we need to have a Dynamic Reference. This dynamic reference object can be detected by all robots. So, for one robot: reliable self-localization => reliable position of the dynamic reference object, which can lead to other robots to self-localize themselves relatively and according to the 'reliable' estimation of the dynamic object. Normally, the dynamic object is the ball. So, "ball localization" is involved also in the process of self-localization.

So to have the cooperation element in the robot team, for Multi Robots: they can all exchange a 'team message'. This message contains the information of: Object Position, Robot ID, Time, and Position Probability. The position probability is an indication of reliability of the dynamic object position estimate. This estimate is to be used as a possible improving information for localization algorithm.



(a) $t=t_1$ (b) $t=t_2$

Calculated Position	Robot ID	Time	Position Possibility
$(2388, 700)$	A	t_1	0.71
$(2264, 658)$	B	t_1	0.92
$(2530, 710)$	E	t_1	0.86
$(2368, 803)$	A	t_2	0.81
$(2401, 801)$	B	t_2	0.91
$(2103, 743)$	C	t_2	0.32
$(2215, 725)$	D	t_2	0.43

In the above figure, the upper-most part represent robot configuration at different time instants. Robots try to localize the ball (round object). The lower table show the exchanged messages for both time instants. You can see that at the first time instant, robots C & D do not see the ball. So, their object estimate is less reliable than the other robots. Robot B can be seen that its estimate of the ball position is more reliable than other robots. This makes robot B the most trusted one. So, further analysis should be done on how to incorporate this information in order for other robots to localize themselves.

Using any algorithm, as explained in other sections of the report, the belief of the location of the robot can be estimated. To include the exchanged information between the team, a further update is applied to the current belief

$$Bel(x_k^a) \leftarrow Bel(x_k^a) \cdot P(x_k^a|obj) \cdot Bel(x_k^b)$$

With $P(x_k^a|obj)$is some evaluation of probability of location of the robot given the dynamic object position coming from the most trusted robot B.

- **RoboCup Teams Localization**

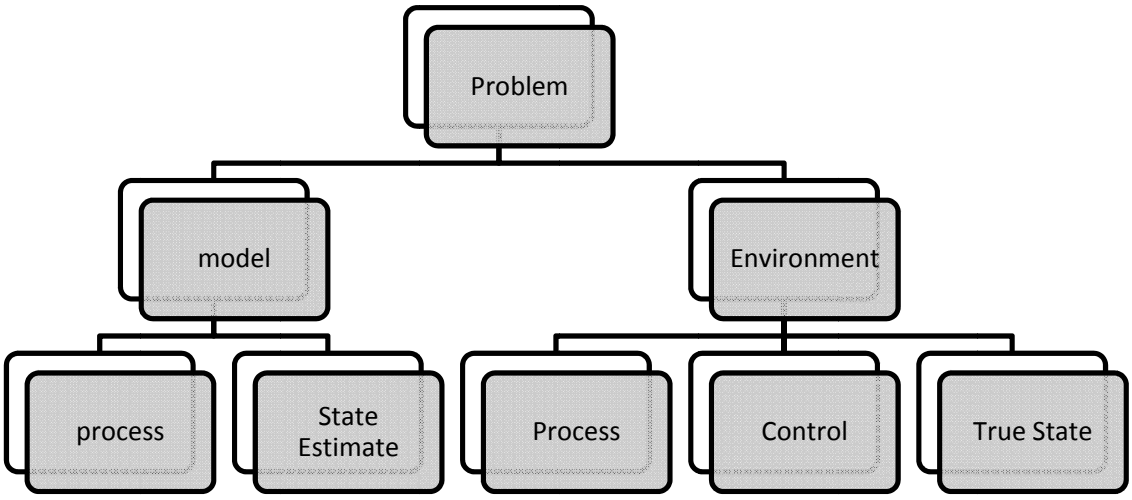
Here is a table of summary for different teams participating in RoboCup competition with their self-localization algorithms.

Team/Univ	Technique	Remarks
Nubots, University of Newcastle, Australia	EKF	When there is insufficient information available or ambiguous, problems with slow estimate 'drift' with time. Some Adaptive Control is done!
BabyTigers DASH, Osaka City University, Japan	MCL	
Cerberus, Bogazici University, Turkey	S-LOC	Mixture Concepts, explained in 2005 report
Eagle Knights, ITAM, Mexico	Triangulation	+ Correction Algorithms
Team Chaos, University of Murcia, Spain	Fuzzy Logic	*claiming* extended techniques with only natural landmarks
S.P.Q.R., Universit`a di Roma, Italy	MCL + SIR	Two stages: MCL then Sampling/Importance Sampling
UChile1, Universidad de Chile, Chile	MCL + EKF	Tried a faster Adaptive-MCL but accuracy low

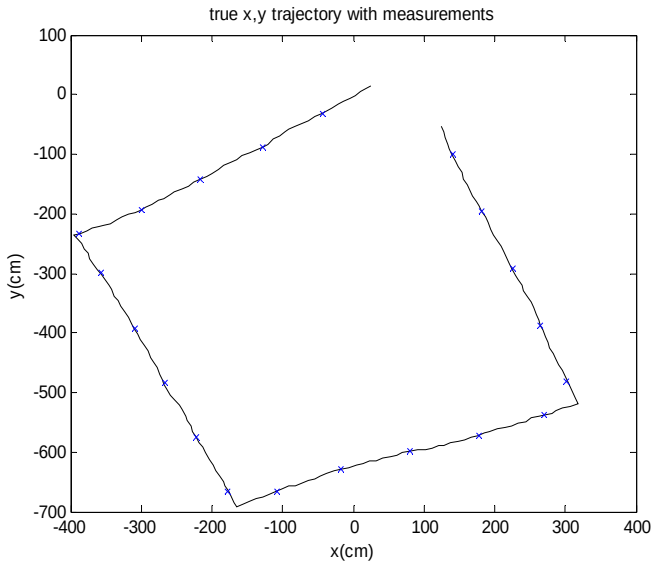
9. Simulation Results

- **Kalman Results**

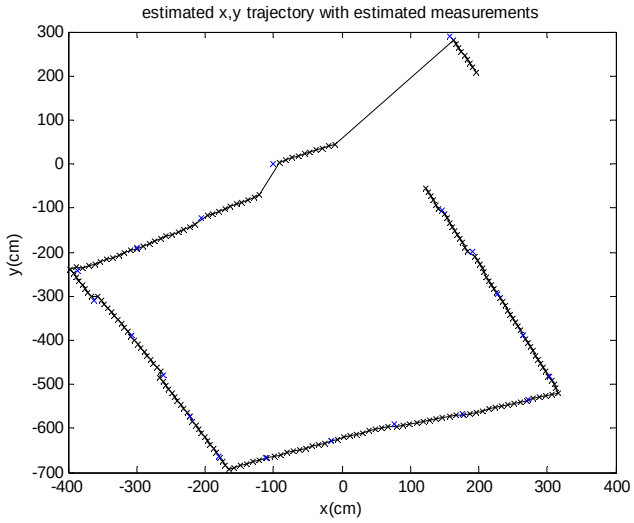
The Kalman Filter example is implemented as shown in the diagram below



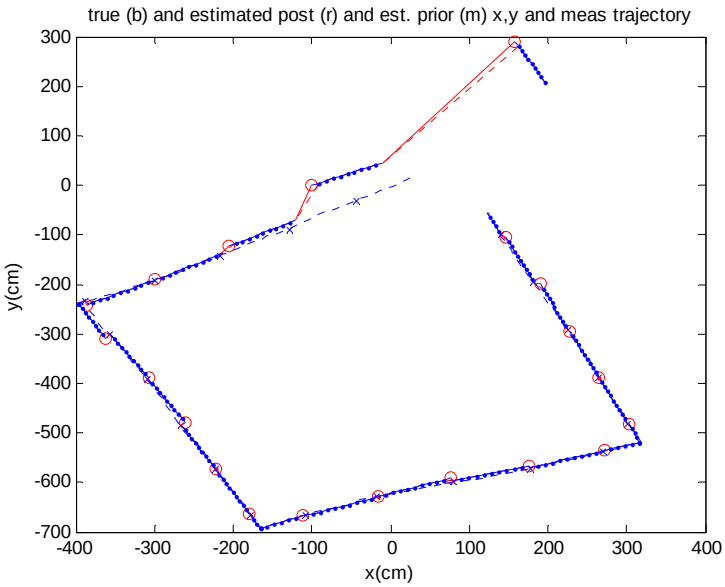
The problem is divided into two submodule, the model describe the Kalman filter process and the estimation results, and the environment describes the true value. If we take have noise with unity covariance we have the following results, the true motion is like the following.



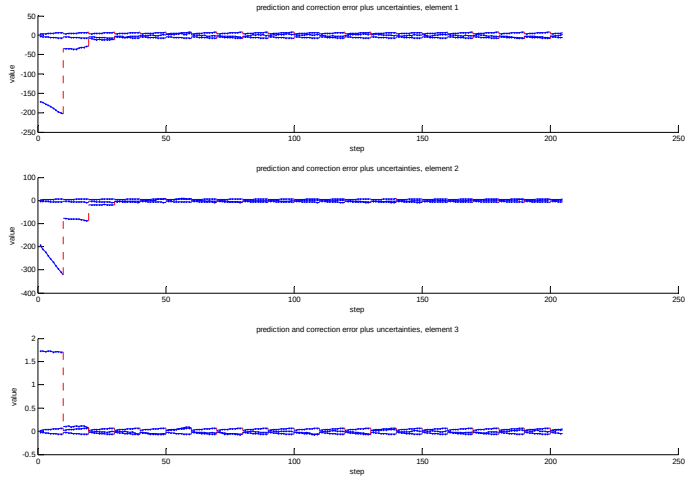
The estimated motion is



We see in the estimation motion a different in the start because we feed the system with wrong initital position. And both the estimated and real together

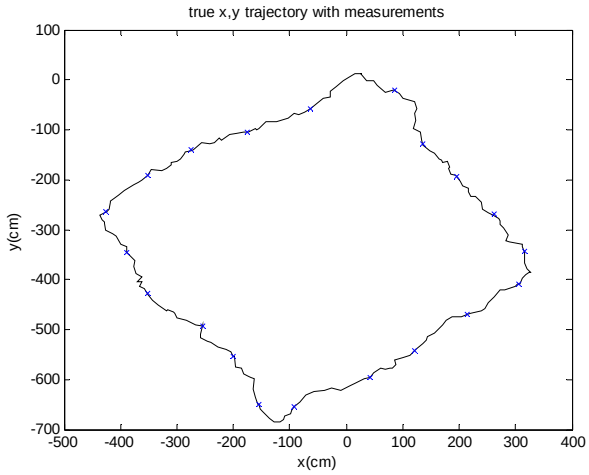


The following diagram shows the error falling between the uncertainty measures of x, y and the orientation

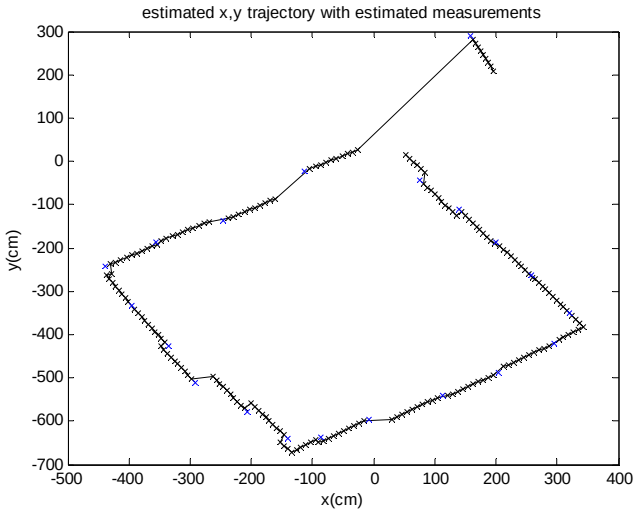


At the beginning the error is high then it approach zero.

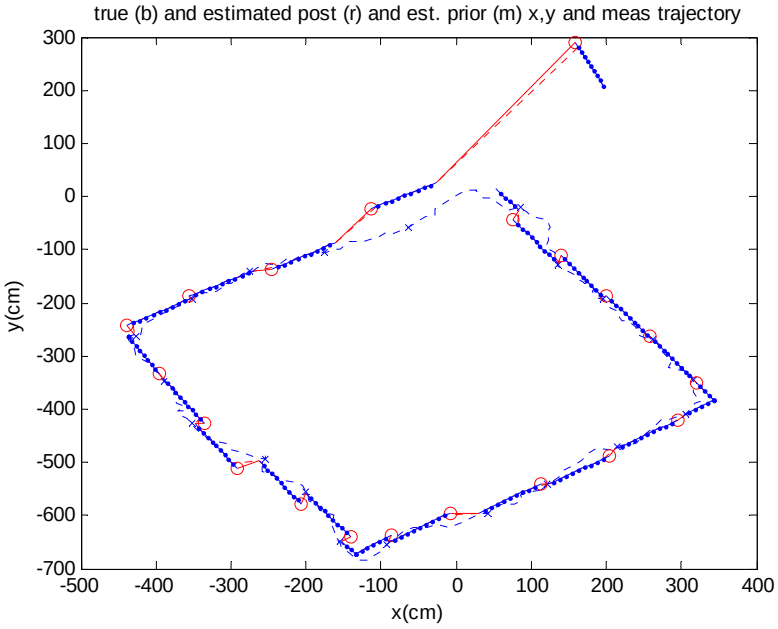
The second case when we increase the uncertainty of the noise on the motion model, then the true movement will be



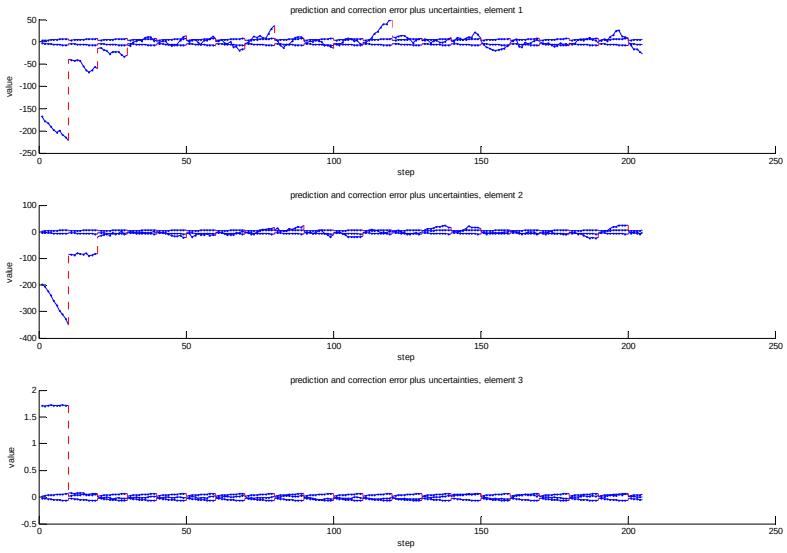
The estimated movement will look like



And both together they will be in this way

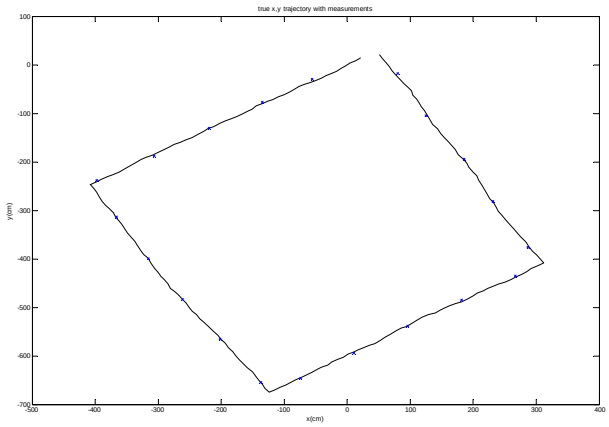


As we can see there is a little difference, this is will be more clear in the next diagram that shows the error

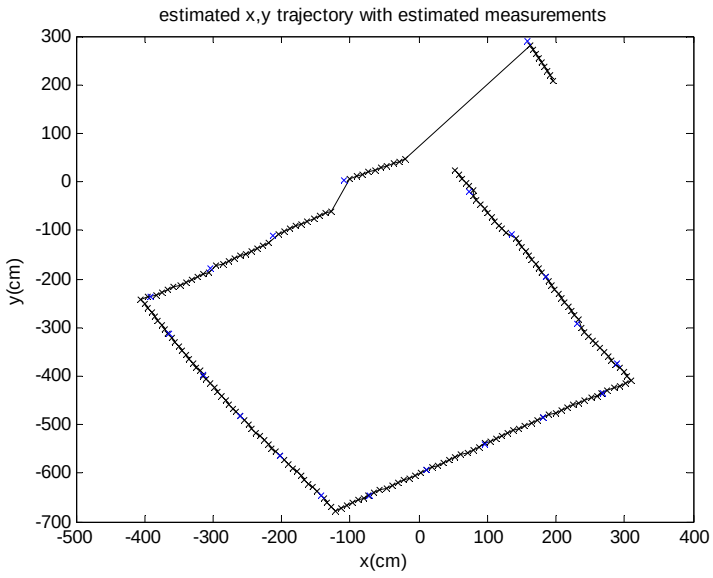


The third experiment when we increase the uncertainty of the error in the process of estimation (measurements)

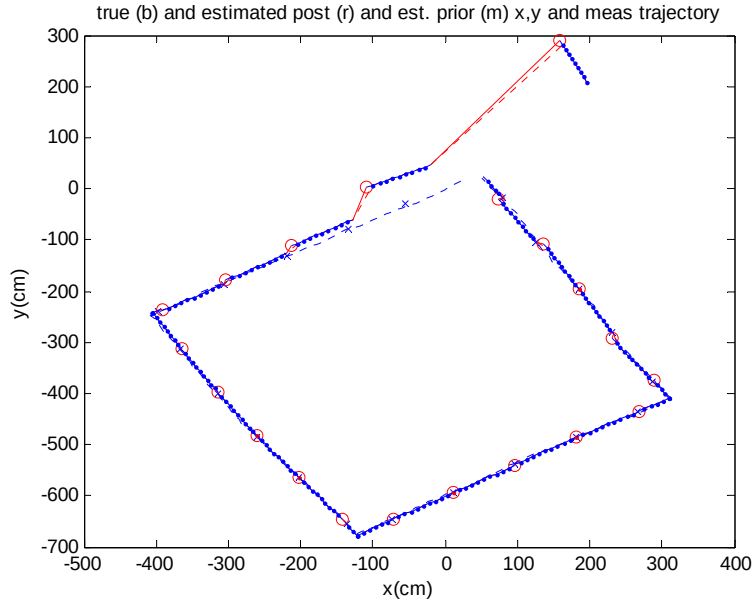
The true path will look like



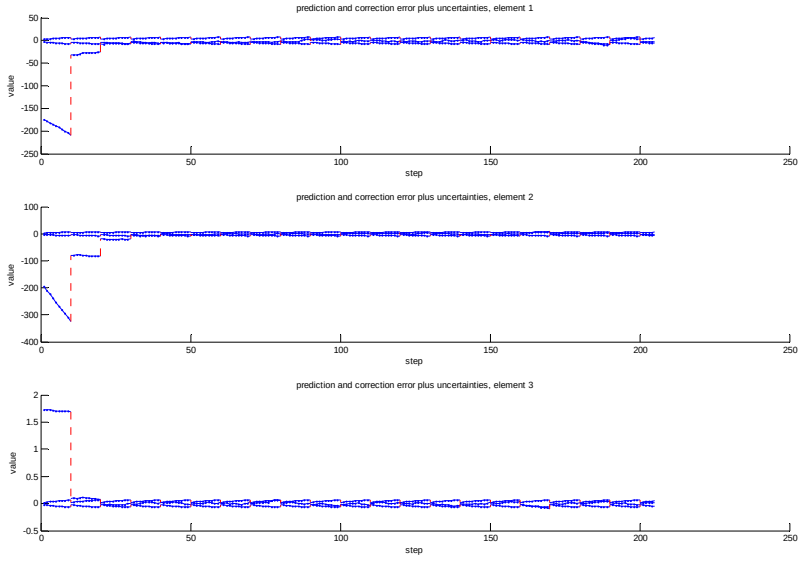
The estimated path will look like the following



And both together are shown below



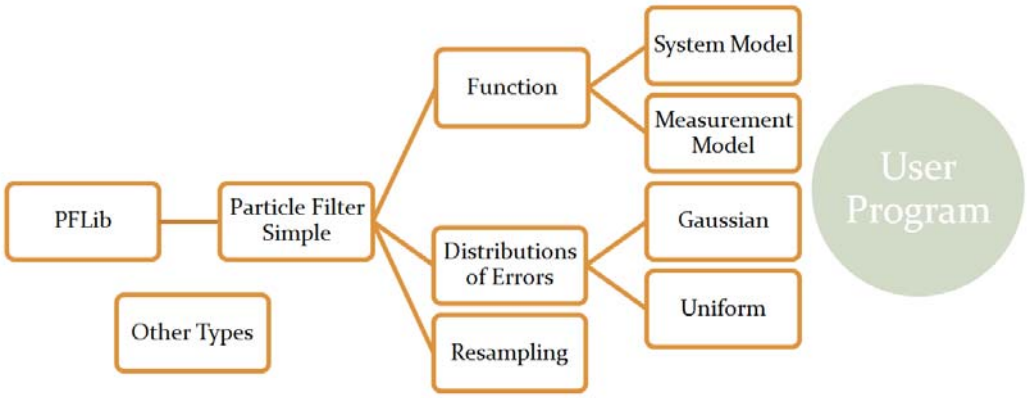
The error measure is shown below



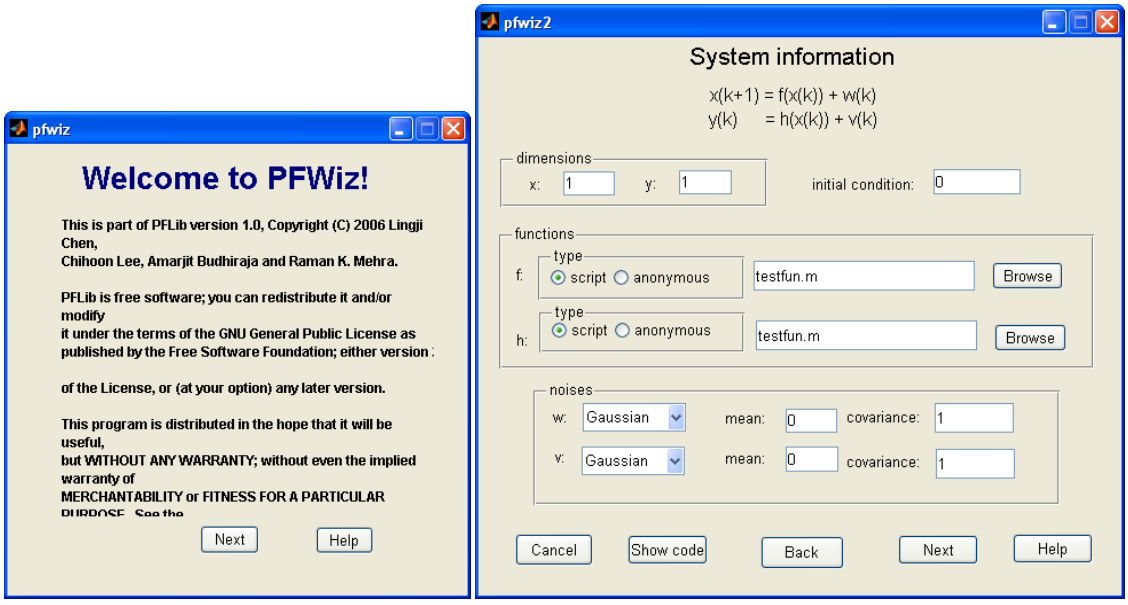
As we can see increasing the error in the measurement model, will not have a significant effect on the estimation process because the Kalman filter will not depend heavily on the measurements because they are not very certain. Kalman filter use lower percentage of measurement model and more of the motion model.

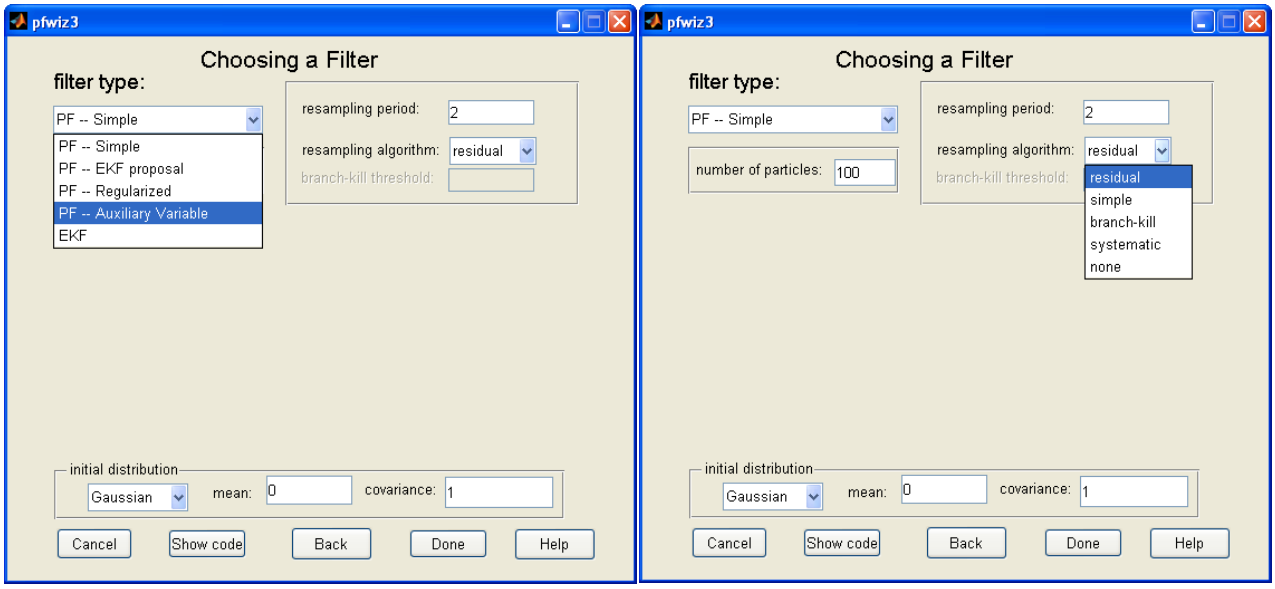
- **Particle Results**

The Particle Filter Library is implemented in the following way as shown in the diagram below

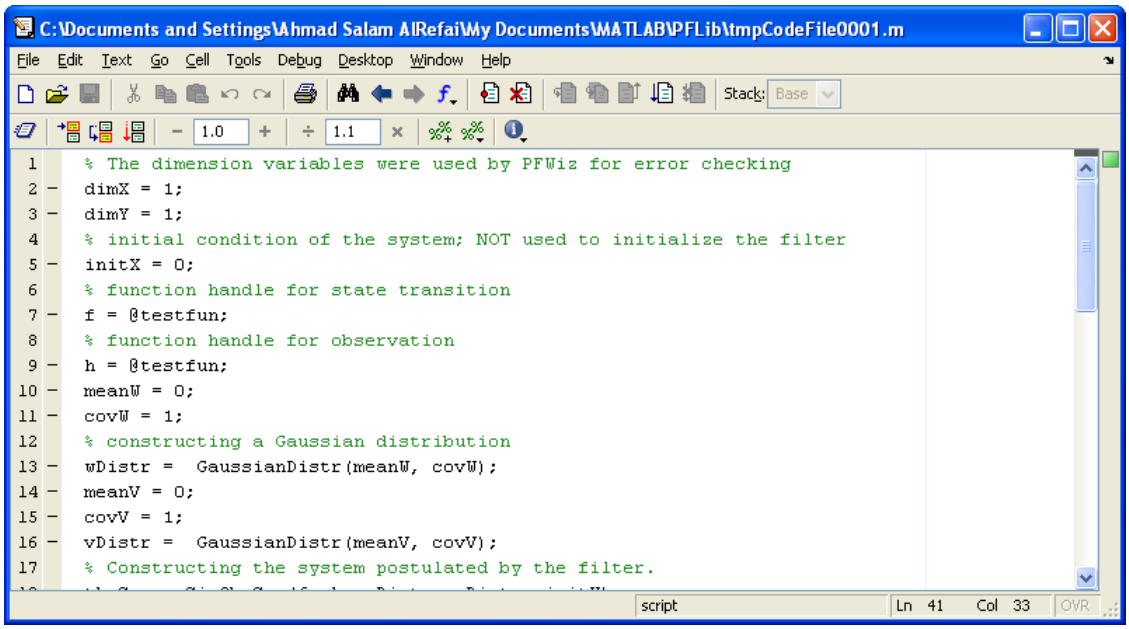


The user program will be able to use function provided in order to estimate the location of the robot. The program provide user interface, they are shown below and are self explanatory…

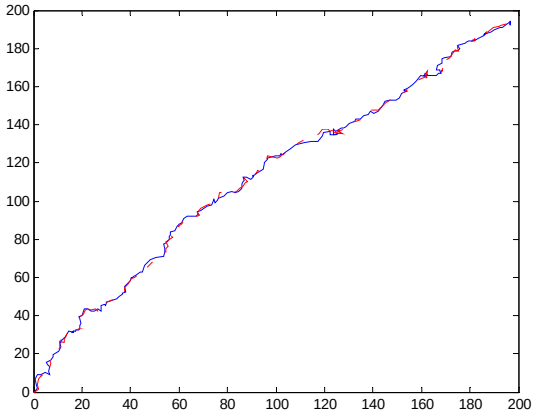




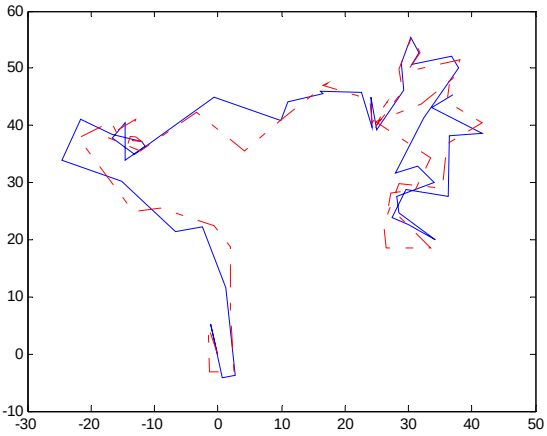
And then the code is generated



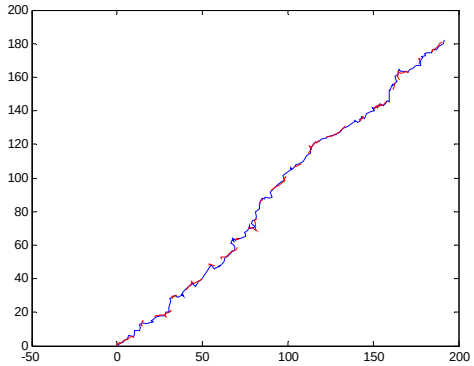
The next diagram show both true and estimated value with identity covariance of the noise in both



When we increase the noise of the true process (motion) the following diagram shows the path



In the above figure we see a large difference between estimated and true value. Next diagram show the true and estimated when we have large varience in the noise in the measurement model


200
180
160
140
120
100
80
60
40
20
0
-50
0
50
100
150
200

10.Future Directions & Discussion

Here, the authors discuss future possible areas of research in order to improve the localization problem solution. So, there are several issues to be discussed:

1) **One big issue** is to study more the efficiency of the techniques. That means to study how to increase the accuracy with keeping computation reasonable. We saw that Kalman filter is computationally cheap. However, it fails to estimate robot location in complex environments with non-Gaussian noise. Particle filter is better that kalman filter because of good results even with no assumptions. However, the computation complexity increases as the number of particles increase which expensive.

   So, a new direction for accurate fast estimation of the belief of the robot location is to study how could the Bayes filter (rule) to help us more to propagate information from previous step to next step with minimum computation and error. Because of the nature of non-Gaussian (multi-modal) dynamically changing environment, completely different approaches should be investigated to make efficient utilization of previous knowledge and current data. The most important issue is to have the previous belief 'transferred' into the future with minimum number of size (e.g. not big size: particles) and with reserved accuracy (e.g. no assumption, so minimum error).

2) **Implementation**: one direct direction to have solid evaluation of the several techniques discussed in to try to implement different scenarios and algorithms into different robotic platforms. The results can be investigated so that further algorithm development can be thought.

11. Team Work

Task	Details
Investigation & Problem Study	**Both** Members from GermanTeam Report 2005
Negative Information	**Shahab** bring the paper, and **Salam** investigate it
Mixture Proposal	**Shahab** bring the paper, and **Both** Study it
Bayes Filtering Techniques	**Salam** bring the paper, and **both** study it
Presentation 1 (Particle Filter Localization)	**Shahab** responsible of Bayes Filter and particle filter and **Salam** present negative information and importance of robotics field.
Kalman Filter Localization Master Thesis	**Salam** bring the thesis, and **both** of authors studied the thesis, **salam** generated simulation results from the code.
Presentation 2(Kalman Filter Localization)	**Salam** introduce the problem, talk about probabilistic framework and Kalman filtering in general. **Shahab** talk specifically about Kalman Filter Localization and the use of landmarks in locating the robot, he also talk about different approached by robocup teams.
PF Library	**Salam** Contacted the professors who developed PFLib Matlab Particle Filter Library and got the code. **Both** Discussed the structure of the code. Salam generated simulation results from the code.
Unscented Kalman/Particle Filter	**Shahab** brings two papers and studied them
Cooperative Localization	**Shahab** Looked at team report who used this technique.
Particle Filter Tutorial	**Salam** brought the paper and studied it
Presentation 3	**Shahab** reminded about Kalman, he explained then Unscented Kalman/Particle Filter, Cooperative Localization. **Salam** talked about particle filtering variations, resampling techniques and simulation of Kalman and Particle results.
Final Report	**Both** wrote the whole report…

12. References

[1] R¨ofer, T., Laue, T., Weber, M., Burkhard, H.D., J¨ungel, M., G¨ohring, D., Hoffmann, J., Altmeyer, B., Krause, T., Spranger, M., Schwiegelshohn, U., Hebbel, M., Nistic´o, W., Czarnetzki, S., Kerkhof, T., Meyer, M., Rohde, C., Schmitz, B., Wachter, M., Wegner, T., Zarges, C., von Stryk, O., Brunn, R., Dassler, M., Kunz, M., Oberlies, T., and, M.R.: GermanTeam RoboCup 2005. Technical report (2005) Available online: http://www.germanteam.org/GT2005.pdf.

[2] Rudy Negenborn. Robot Localization and Kalman Filters: On finding your position in a noisy world. Master Thesis 2003, Utrecht University.

[3]Arulampalam, S., Maskell, S., Gordon, N., Clapp, T.: A Tutorial on Particle Filters for On-line Non-linear/Non-Gaussian Bayesian Tracking.

[4]Rekleitis, I.M.,: A Particle Filter Tutorial for Mobile Robot Localization.

[5]Fox, D., Burgard, W., Dallaert, F., Thrun, S.,: Monte Carlo Localization: Efficient Position Estimation for Mobile Robots. Proceedings of the 16th National Conf. on A.I. 1999.

[6]Wan, E., v.d.Merwe, R.,: The Unscented Kalman Filter for Nonlinear Estimation.

[7]Wang, F., Zhao, Q., Deng, H.,: A Mixed Fast Particle Filter. 8th Int'l Conf. on Software Engineering, A.I., Networking and Parallel/Distributed Computing 2007.

[8]Thrun,S., Fox, D., Burgrad, W.,: Monte Carlo Localization with Mixture Proposal Distribution. AAAI 2000.

[9] Jan Hoffmann, Michael Spranger, Daniel G¨ohring, and Matthias J¨ungel,: Making Use Of What You Don't See: Negative Information In Markov Localization.

[10] Qining Wang, Chunxia Rong, Lianghuan Liu, Hua Li, and Guangming Xie,: The 2006 sharPKUngfu Team Report. Peking University, China.