CSE 510 – ROBOTICS ALGORITHMS

LOCALIZATION: PARTICLE FILTERING AND KALMAN FILTERING AND THEIR COMPARISON

Surya Kumar Selvam

UB# 50026556

suryakum@buffalo.edu

INTRODUCTION:

Robot Localization is one of the key problems in making autonomous robots. While the GPS can give the position of an object, the localization scale of GPS is very large (up to 10 m) which is not plausible in the real world robot. So there must be some methods for the robot to completely localize itself. In order to localize itself, a robot has access to relative and absolute measurements giving the robot feedback about its driving actions and the situation of the environment around the robot. Given this information, the robot has to determine its location as accurately as possible. What makes this difficult is the existence of uncertainty in both the driving and the sensing of the robot. The uncertain information needs to be combined in an optimal way.

LOCALIZATION:

Knowing your location, and being able to navigate to other locations, is extremely important for autonomous robots. The act of finding one's location against a map is known as localization. The most common form of intelligent localization and navigation is to use a map, combined with sensor readings and some form of closed-loop motion feedback. But a robot needs to do so much more than just localizing itself. Often, we'd like our robots to be able to build their own maps, since map building by hand is tedious, boring, and error-prone. The field of robotics that studies localization and mapping is typically called SLAM (simultaneous localization and mapping).

Techniques for localization

Few of the techniques used for localization are

i) **Bayesian sequential Filter**

This is a localization technique used when the movement of the robot is discrete. A Bayes filter is an algorithm used in computer science for calculating the probabilities of multiple beliefs to allow a robot to infer its position and orientation. Essentially, Bayes filters allow robots to continuously update their most likely position within a coordinate system, based on the most recently acquired sensor data. This is a recursive algorithm. It consists of two parts: prediction and innovation.

ii) **Kalman Filter**

The Kalman filter is a set of mathematical equations that provides an efficient computational (recursive) means to estimate the state of a process, in a way that minimizes the mean of the squared error. The filter is very powerful in several aspects: it supports estimations of past, present, and even future states, and it can do so even when the precise nature of the modeled system is unknown. The Kalman filter addresses the general problem of trying to estimate the state of a Discrete-time controlled process that is governed by the linear stochastic difference equation. With a measurement

iii) **Particle Filter**

Particle filtering is a general Monte Carlo (sampling) method for performing inference in state-space models where the state of a system evolves in time and information about the state is obtained via noisy measurements made at each time step. It contains creation of N number of particles in the environement and then converge to the particles which are near the robot or similar to the robot.

Project Description

The Project mainly involves the localization and tracking of a robot in a 2-D Planar environment using Kalman Filtering as well as Particle filtering and compare them. The measurement available for the system is the laser which can measure distance to the obstacle directly in front of it. The state of a robot at any instant is given by (x, y, theta) for both the filters.

The state space in Kalman filtering is continuous, uni-modal. When it comes to scaling, Kalman filtering is quadratic approximate filter.

Particle filter is a multimodal, continuous filter which is highly computing intensive but it scales much better than any other filtering methods in higher dimensions at the cost of some computational task.

Logic and Approach Used

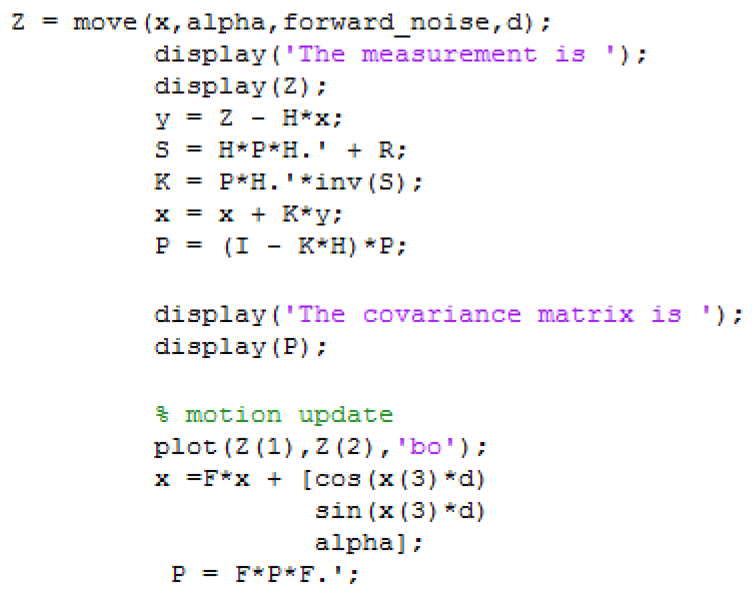
Kalman Filtering

The Kalman filtering mainly consists of two parts, the measurement model and the update or observation model. The measurement model can be best described by the state transition process. The state transition I used for this Kalman Filtering is

$$\begin{bmatrix} X_{t+1} \\ Y_{t+1} \\ \Theta_{t+1} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 1 \end{bmatrix} * \begin{bmatrix} X_t \\ Y_t \\ \theta_t \end{bmatrix} + \begin{bmatrix} \cos\theta & 0 \\ \sin\theta & 0 \\ 0 & 1 \end{bmatrix} * \begin{bmatrix} d \\ \alpha \end{bmatrix}$$

In my problem, at each step the measurement is done by moving in a steering angle alpha with a Gaussian noise of 0 mean and 0.05 variance. After using the error in measurement and prediction, the covariance matrix P is updated with the Kalman gain found.

Then the model is updated with the predicted x and the u vector which is the second part of the above equation. This method is simulated for 10 time steps before the prediction almost exactly approximates the measurement model.



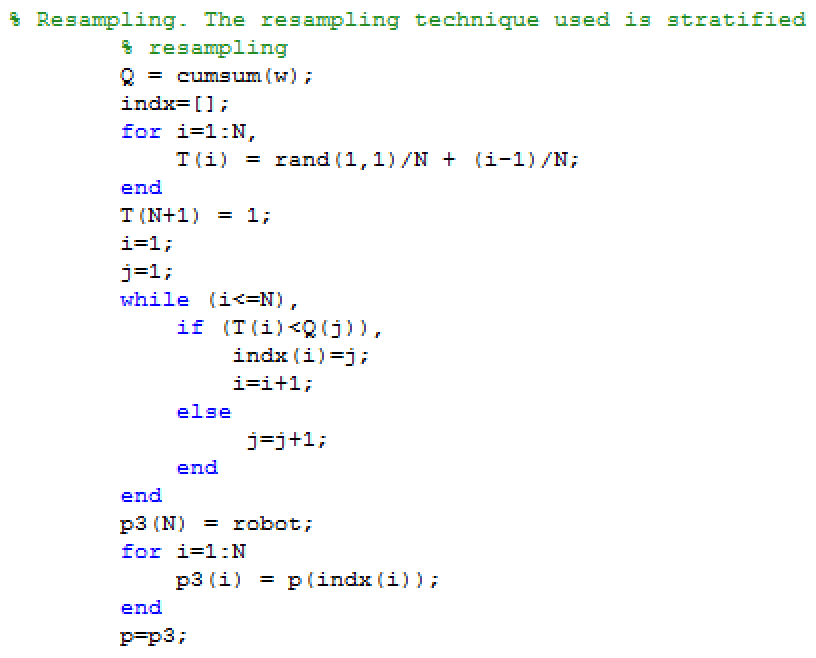
```
Z = move(x,alpha,forward_noise,d);
         display('The measurement is ');
         display(Z);
         y = Z - H*x;
         S = H*P*H.' + R;
         K = P*H.'*inv(S);
         x = x + K*y;
         P = (I - K*H)*P;

         display('The covariance matrix is ');
         display(P);

         % motion update
         plot(Z(1),Z(2),'bo');
         x =F*x + [cos(x(3)*d)
                   sin(x(3)*d)
                   alpha];
          P = F*P*F.';
```

Particle Filtering

In Particle filtering I created N= 1000 number of particles. The technique used here is all the particles are moved in the same direction as the robot for each time step. The particles which signifies approximately the robot achieve greater gain and are taken to the next iteration. I used stratified resampling technique to select the particles with maximum importance weight. The robot.m class contains all the functions which are necessary for processing the particles.



```
% Resampling. The resampling technique used is stratified
        % resampling
        Q = cumsum(w);
        indx=[];
        for i=1:N,
            T(i) = rand(1,1)/N + (i-1)/N;
        end
        T(N+1) = 1;
        i=1;
        j=1;
        while (i<=N),
            if (T(i)<Q(j)),
                indx(i)=j;
                i=i+1;
            else
                 j=j+1;
            end
        end
        p3(N) = robot;
        for i=1:N
            p3(i) = p(indx(i));
        end
        p=p3;
```

Simulation

Simulation Language: Mat lab

Program Used : kf.m for Kalman filtering and robot.m and temp.m for Particle filtering

Configurations:

1) Kalman Filtering

a) Initial Position = (X, Y, Orientation) = (1,1,0)

b) Initial Covariance = eye(3) * 10000

c) Forward noise = Gaussian with 0 mean and 0.05 variance

d) Number of time steps = 10

e) Distance for each step = 1

2) Particle Filtering

a) Forward noise = 0 mean with 0.1 variance

b) Sense noise = 0 mean with 0.1 variance

c) Turn noise = 0

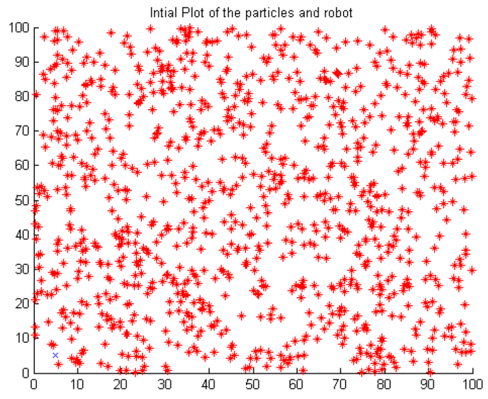
d) Initial Config of Robot = (5, 5, 0)

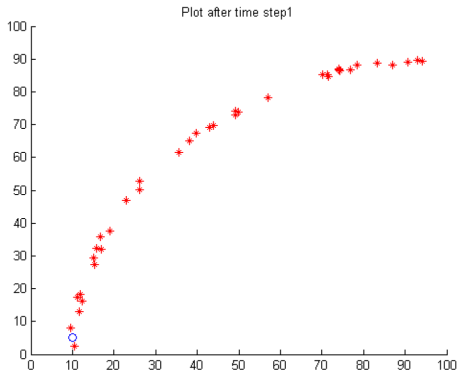
e) No of Time steps = 15

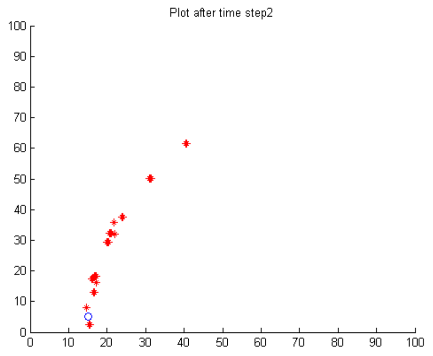
f) No of Particles Used = 1000

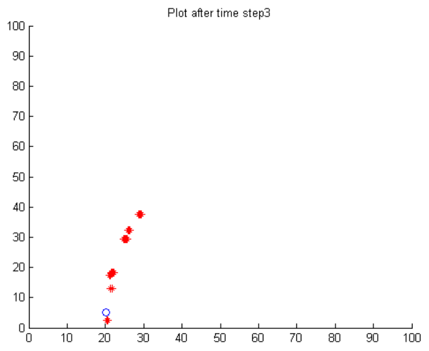
Results and Plots

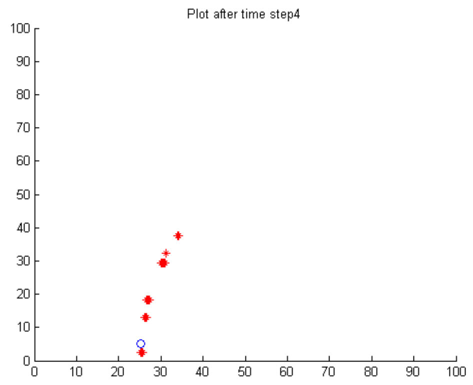
Particle Filter (Step Wise)

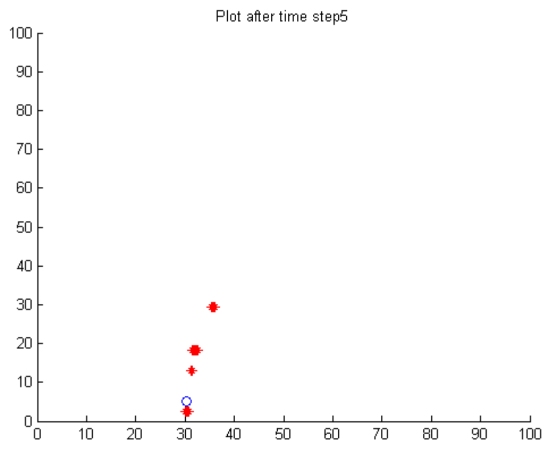


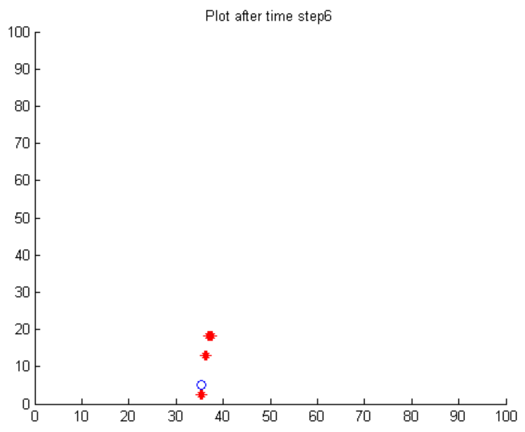


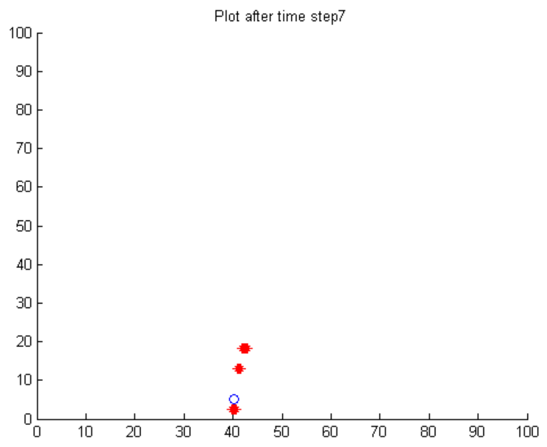


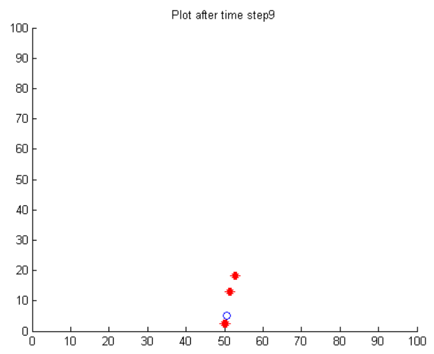


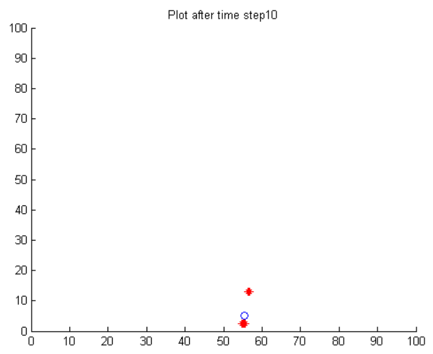

Plot after time step4

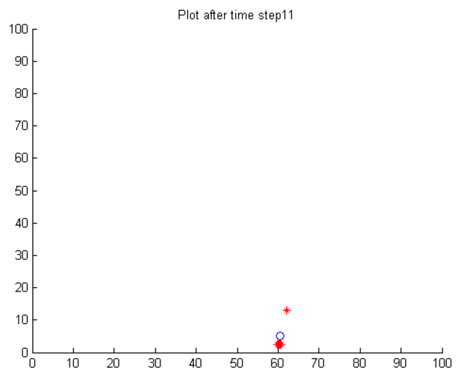

Plot after time step5

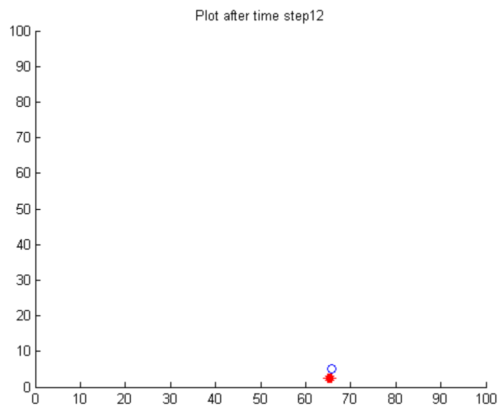

Plot after time step6

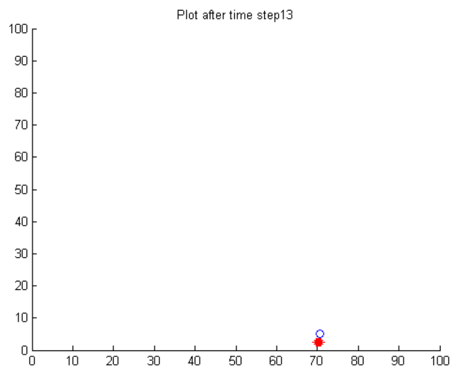

Plot after time step7

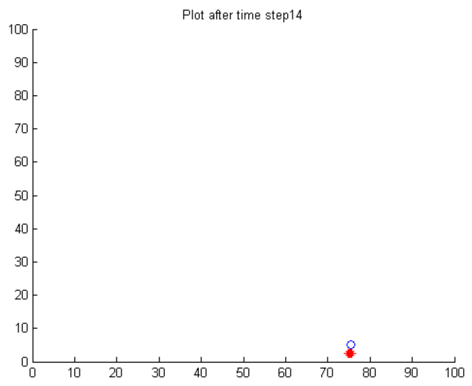

Plot after time step9

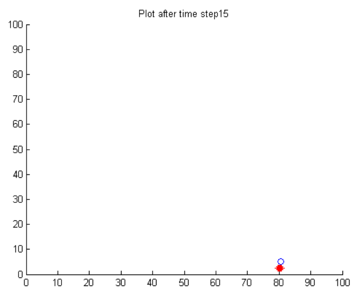

Plot after time step10


Plot after time step11

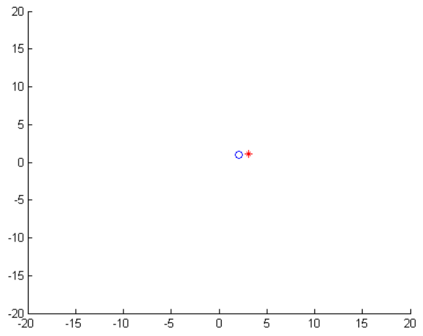
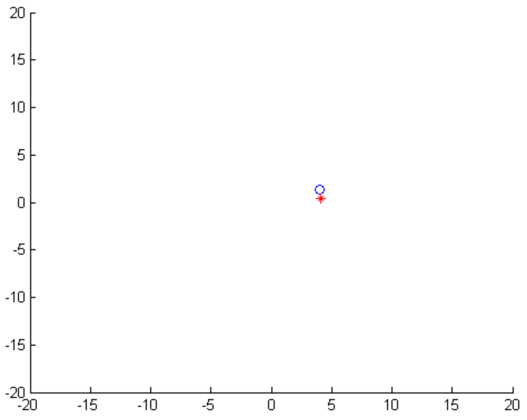
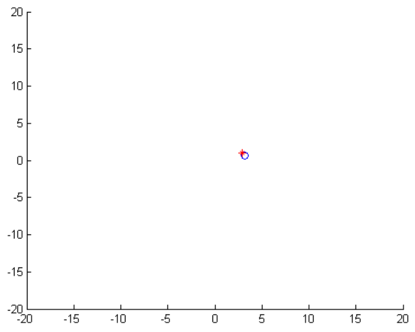
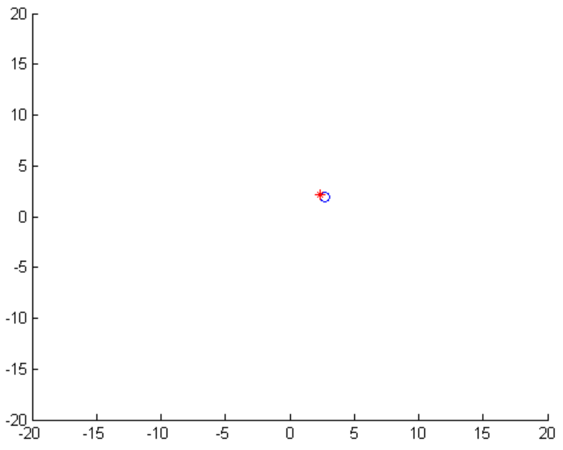
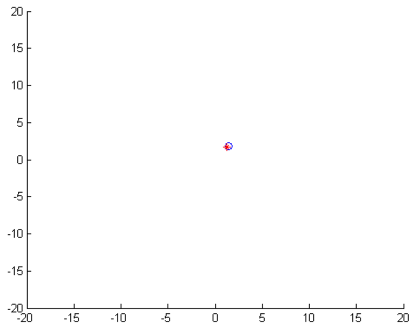
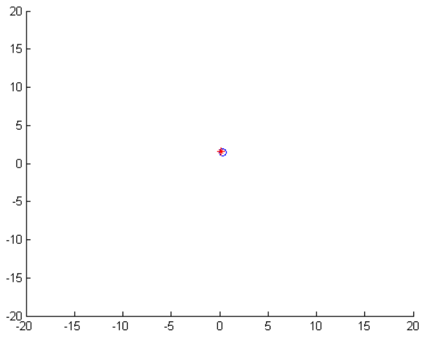

Plot after time step12

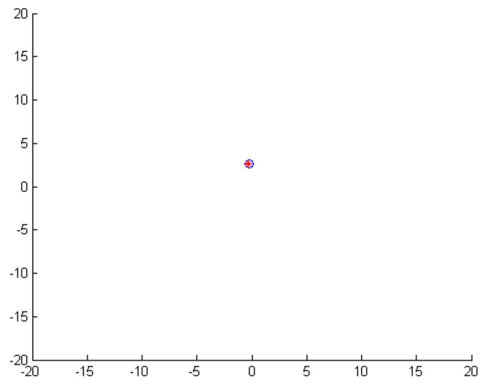

Plot after time step13

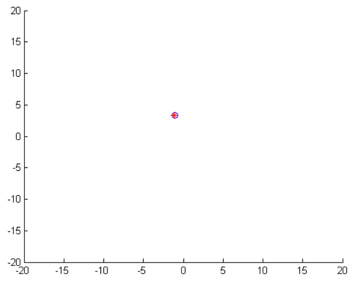

Plot after time step14

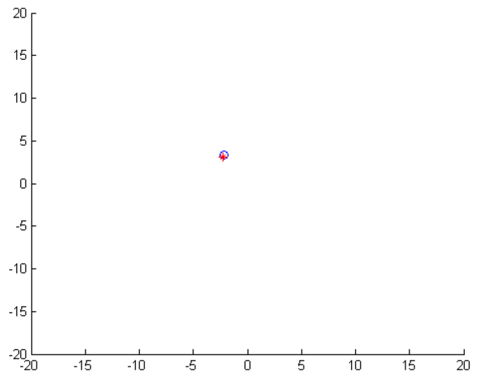

Plot after time step15

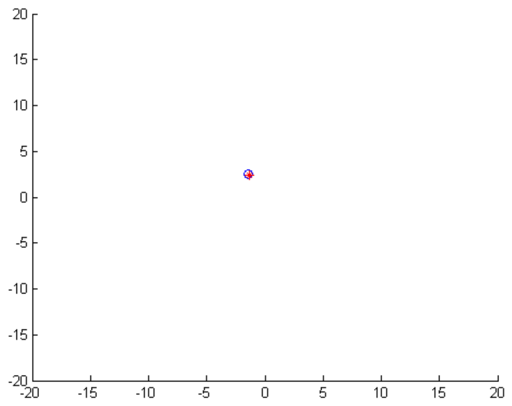
Kalman Plots









Observation and Conclusion

In a linear system with Gaussian noise, the Kalman filter is optimal. In a system that is nonlinear, the Kalman filter can be used for state estimation, but the particle filter may give better results at the price of additional computational effort. In a system that has non-Gaussian noise, the Kalman filter is the optimal *linear* filter, but again the particle filter may perform better.

While the Kalman Filter performed admirably well in the experiment (It almost converged as early as the 4th step), the Particle filter was computationally intensive. But even though it is intensive, particle filters perform supposedly well in higher dimension localization. Also, programming particle filter was easy when we compared to Kalman filters where lots of logics are used.

Both the filters were able to localize well within the time steps and accurately.

Bibliography

http://cs.stanford.edu/groups/manips/teaching/cs223a/ - main source

http://www.cs.unc.edu/~welch/media/pdf/kalman_intro.pdf

http://www.cs.ucf.edu/~mikel/Research/tutorials/kalman-filters-a-tutorial.pdf

http://robots.stanford.edu/papers/thrun.pf-in-robotics-uai02.pdf - Detailed explanation of Particle Filtering

http://www.bcs.rochester.edu/people/eorhan/notes/particle-filtering.pdf