NORTHWESTERN
UNIVERSITY

Electrical Engineering and Computer Science Department

SAJID ULLAH

Technical Report
Number: NU-EECS-16-06

December, 2015

"ON THE USE OF KALMAN, AND PARTICLE FILTERING FOR POSITIONING, NAVIGATION AND TRACKING IN AUTONOMOUS DRIVING VEHICLE"

Courtesy of Stanford university

PARTICLE FILTERING HAS APPLICATION IN DOMAIN SUCH AS COMPUTER GRAPHICS, AI, FINANCE AND BUSINESS

Kalman and particle filtering are so viable to Stanley self-driving car that it won DARPA Grand Challenge , a 212 km (132 mi). The two filtering technique are used for localization, and tracking of object in real world. This work is inspired by Stanley and Sebastian Thrun.

ABSTRACT

Autonomous driving vehicle (ADV) used GPS for its positioning (localization) while navigating. This work includes simulation of positioning, localization, and tracking. The radio signal of GPS is subject to time delay, obstacle as tunneling, and multipath propagation and other means of signal deteriorating phenomenon.so to overcome these problem, the robot has supplement information from sensor it deploys i.e. wheel speed sensor, accelerometer, and Gyroscope and inertial system. The data from sensor are fused and used to predict the future position of vehicle. This data is constantly checked with the GPS data.so fusing together positioning and Map-aided positioning algorithm the absolute position of the vehicle is made consistent with digital Map. This survey study will cover problems

I. Kalman filtering and Map Matching in localization
II. Monto Carlo Simulation methods for object tracking. The estimation technique will use particle filtering for Bayesian estimation

In Simulation results, we track a red ball and red box. The code was run on core I-3, 2.10 GHz processor, and coding was done in Matlab. Linear tracking of path was done by Matlab toolbox KalmanAll.

A. MAP MATCHING

Map matching is a technique in GIS that associates a sorted list of user or vehicle positions to the road network on a digital map. The main purposes are to track vehicles, analyze traffic flow and finding the start point of the driving directions. In real world, driver uses GPS for tracking. The precision of GPS is lower in area where the radio signals are hindered by tall building, through the tunnel and large forestry. GPS can be subject to sampling error. To tackle these problem an algorithm is needed to be implemented. An open source prototype for map matching is implemented with the help of the routing engine GraphHooper in Java.

GraphHooper provide the following function for ADV

I. Routing
II. The route optimization problem
III. Self-hosting
IV. The Geocoding API
V. The matrix API

Routing is the process of finding the 'best' path(s) between two or more points, where best depends on the vehicle and use case. With our API you have a fast and solid way to find this best path. The matrix API find the shortest distance in a matrix

The survey study of Stanley and junior robot is summarized in [3], [4]. Modern day vehicle has driver support system. Dead reckoning (DR) can suffer from error accumulation and GPS masking can occur the GPS can be prone to error, so to help out the device to localize the ADV in map matching we use dead reckoning and integrate the information. Dead reckoning is relative easy way(computationally) easy way to calculate at any given time. The initial position and all previous displacement are added in the final equation. DR system uses several methods to calculate. A common approach is to calculate speed and angular velocity (yaw rate and pitch rate). The error due to DR’s inaccuracies accumulate over the time. For compensate for these error, we integrate information from GPS and DR.to achieve, Kalman filter is used.

A.1. Kalman filter simulation:

Code:

```
% Make a point move in the 2D plane
% State = (x y xdot ydot). We only observe (x y).
% X(t+1) = F X(t) + noise(Q)
% Y(t) = H X(t) + noise(R)
ss = 4; % state size
os = 2; % observation size
F = [1 0 1 0; 0 1 0 1; 0 0 1 0; 0 0 0 1];
H = [1 0 0 0; 0 1 0 0];
Q = 0.1*eye(ss);
R = 1*eye(os);
initx = [10 10 1 0]';
initV = 10*eye(ss);
seed = 9;
rand('state', seed);
randn('state', seed);
T = 15;
[x,y] = sample_lds(F, H, Q, R, initx, T);
[xfilt, Vfilt, VVfilt, loglik] = kalman_filter(y, F, H, Q, R, initx, initV);
[xsmooth, Vsmooth] = kalman_smoother(y, F, H, Q, R, initx, initV);
dfilt = x([1 2],:) - xfilt([1 2],:);
mse_filt = sqrt(sum(sum(dfilt.^2)))
dsmooth = x([1 2],:) - xsmooth([1 2],:);
mse_smooth = sqrt(sum(sum(dsmooth.^2)))
subplot(2,1,1)
hold on
plot(x(1,:), x(2,:), 'ks-');
plot(y(1,:), y(2,:), 'g*');
plot(xfilt(1,:), xfilt(2,:), 'rx:');
for t=1:T, plotgauss2d(xfilt(1:2,t), Vfilt(1:2, 1:2, t), 1); end
hold off
legend('true', 'observed', 'filtered', 0)
xlabel('X1')
ylabel('X2')
subplot(2,1,2)
hold on
plot(x(1,:), x(2,:), 'ks-');
plot(y(1,:), y(2,:), 'g*');
plot(xsmooth(1,:), xsmooth(2,:), 'rx:');
for t=1:T, plotgauss2d(xsmooth(1:2,t), Vsmooth(1:2, 1:2, t), 1); end
hold off
legend('true', 'observed', 'smoothed', 0)
xlabel('X1')
ylabel('X2')
```

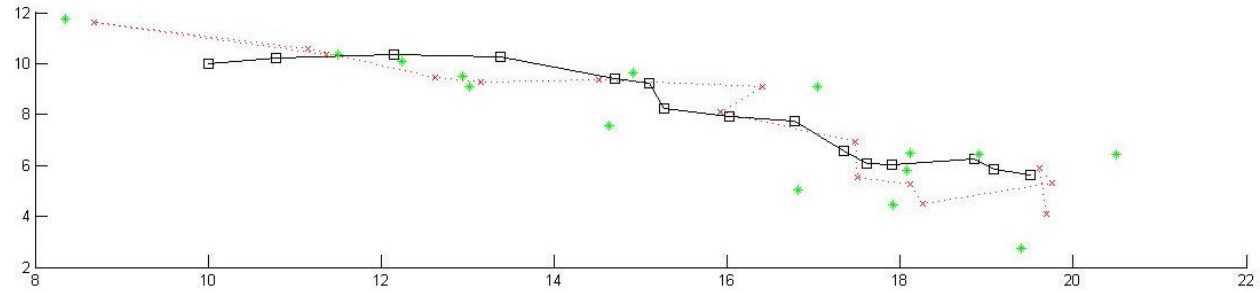


Fig 1. estimation of point position using Kalman filter (Dead reckoning)

Consider a particle moves in the plane at constant velocity. the position of the body changes from old position to new one.

New position =Old position +changes in displacement+ Noise

The starting position is (10,10 moving toward the right with (1,0). Below are observed and filtered estimation of position. The mean and square error of the filtered estimate is 4.9 while smooth estimate is 3.2.

This result was obtained by Matlab toolbox ‘KalmanAll’

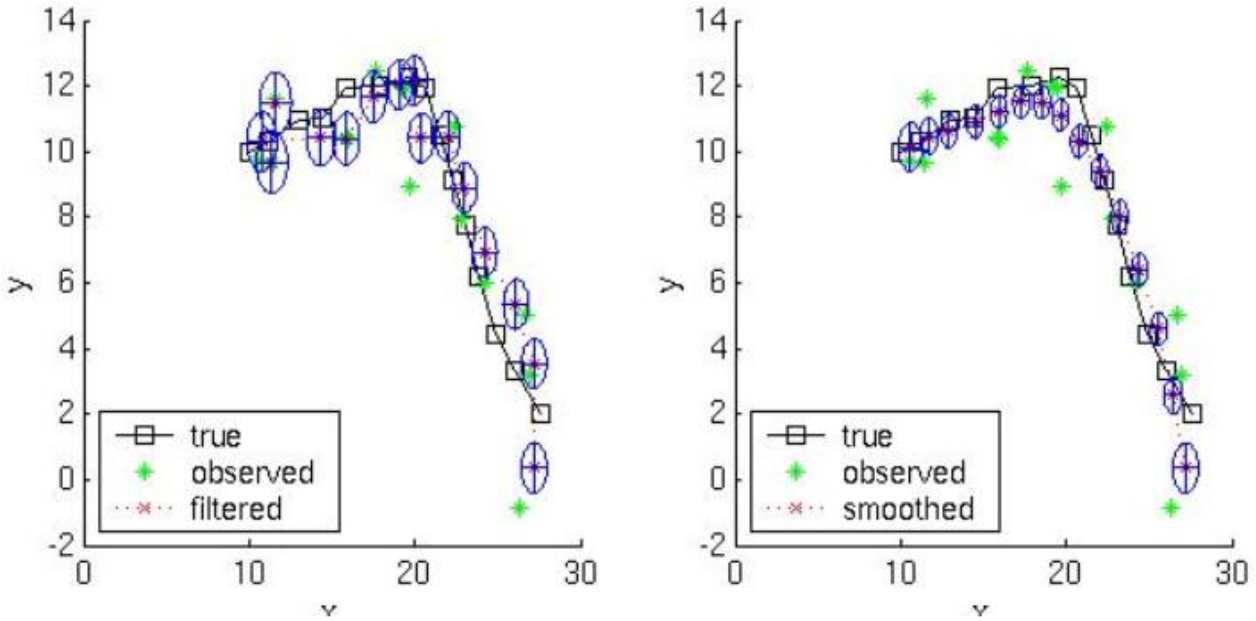


Fig 1. (b) localization of an object

Commercial available map provide the standard with which the car matches its GPS coordinates

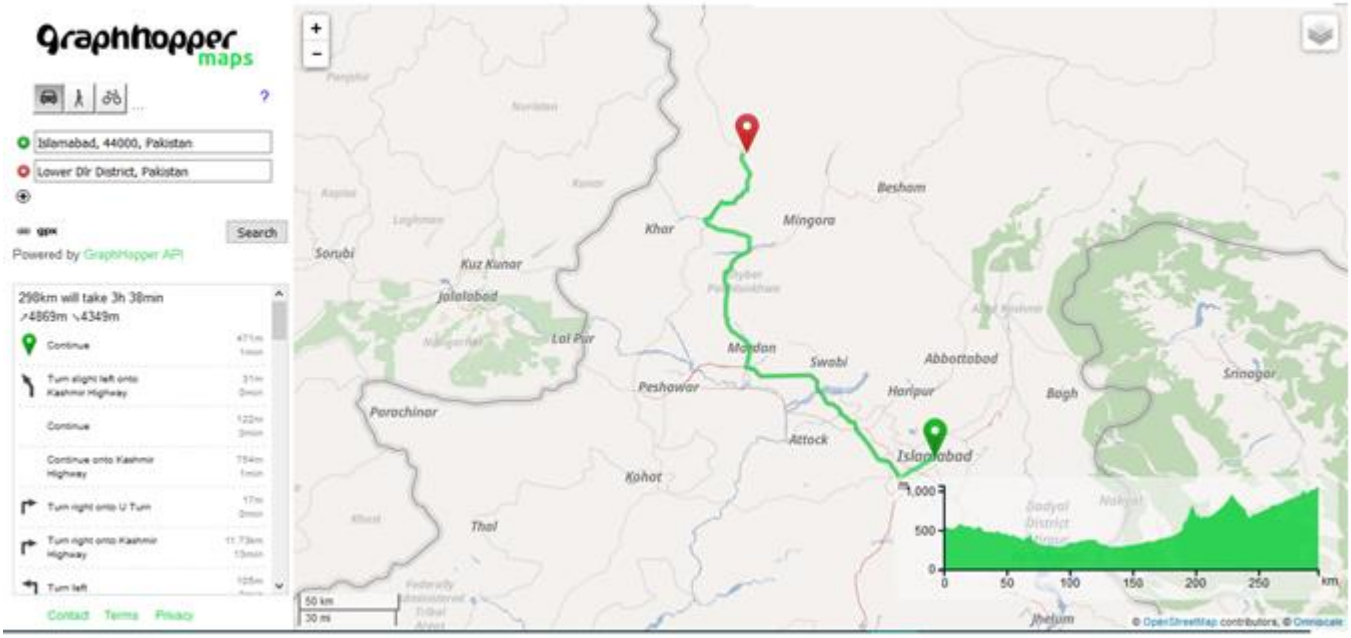


Fig 2. GraphHooper provide distance matrix, used by autonomous driving vehicle industry in localization

A.2. Motion Model:

When we track an object, we use motion model for it. The variable of interest in object tracking are given in the table. In addition to that other state variable necessary to ADV is given in the table.

Table 1. variables used in ADV

No of values	State variable
3	Position (longitude, latitude, altitude)
3	Velocity
3	Orientation (Euler angles: roll, pitch, and yaw)
3	Accelerometer biases
3	Gyro biases

B. Monte Carlo Methods:

Reminding once more, the report is based on Junior autonomous vehicle. Tracking of vehicle and other world object is done by laser range finder. The algorithm used combine geometrical and dynamic properties from the environment, and estimation is done using Bayes filter. Single Bayes filter is used for every vehicle. junior is supposed to maneuver in urban environment, which demand static and dynamic awareness. This approach handle data segmentation and association. The laser ranger finder has frame rate of 10 Hz, and average update rate of 40 Hz. Monte Carlo Simulation work is summarized in [1].

Multiple vehicles are tracked when they are closed enough and Bayesian filter is applied for each vehicle. A Mathematical tractable model is used joint probability of state parameter of all vehicle.

B.1. Particle Filtering:

Monte- Carlo method is generalized term for recursive and evolutionary data processing algorithm. Particle filtering is alternative nonparametric implementation of the Bayes filter [5]. Particle filters approximate the posterior by a finite number of parameters. Particle filtering approximate posterior probability by number of parameter. Particle represents state of individual sample of posterior distribution. These state in fact represent hypothesis, which is multiple hypothesis testing.

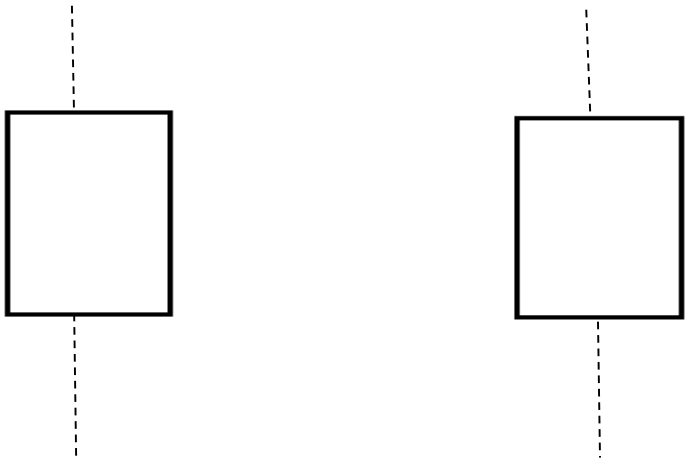


Fig 3. Multiple car on lane

1: **Algorithm Particle_filter**($\mathcal{X}_{t-1}, u_t, z_t$)**:**
2: $\bar{\mathcal{X}}_t = \mathcal{X}_t = \emptyset$
3: for $m = 1$ to M do
4: sample $x_t^{[m]} \sim p(x_t \mid u_t, x_{t-1}^{[m]})$
5: $w_t^{[m]} = p(z_t \mid x_t^{[m]})$
6: $\bar{\mathcal{X}}_t = \bar{\mathcal{X}}_t + \langle x_t^{[m]}, w_t^{[m]} \rangle$
7: endfor
8: for $m = 1$ to M do
9: draw i with probability $\propto w_t^{[i]}$
10: add $x_t^{[i]}$ to $\mathcal{X}_t$
11: endfor
12: return $\mathcal{X}_t$

Fig 4. particle filtering algorithm

The purpose of particle filtering is to generate sample that follow filter distribution.one can estimate the state by these methods

I. Using mean or median
II. Using empirical distribution

The algorithm of particle filtering has two steps.one is prediction and the other is filtering.

- Prediction is time evolution of particle according to system model
- Reselection particle filtering particle according to their likelihood

The states are location and speed of red object and the observables are color of pixel on which the particle exists. The observations are estimated and filtered which are close

Consider State space model, which takes signal X

$$\begin{array}{ccccccccc} X_0 & \rightarrow & X_1 & \rightarrow & X_2 & \rightarrow & X_3 & \rightarrow & \dots \quad \text{signal} \\ \downarrow & & \downarrow & & \downarrow & & \downarrow & & \dots \\ Y_0 & & Y_1 & & Y_2 & & Y_3 & & \dots \quad \text{observation} \end{array}$$

In tracking object in video frame the red pixel when tracking red object will be our observables.

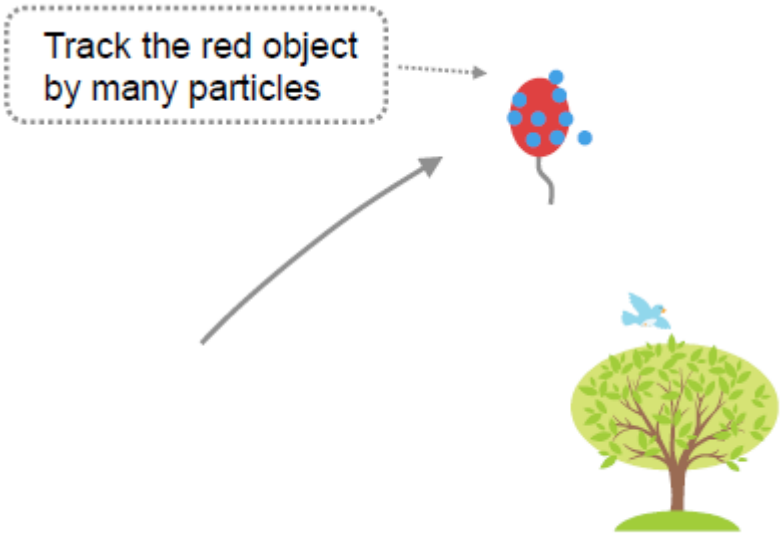


Fig 5. single position and velocity state of red object estimated by multiple particles

X is three dimensional vector

The update equitation used in particle filtering are given as,

$$X_n = \begin{pmatrix} X_n^{(1)} \\ X_n^{(2)} \\ X_n^{(3)} \end{pmatrix}$$

$$\begin{cases} X_n^{(1)} &= X_{n-1}^{(1)} + \epsilon_n \; W_n \\ X_n^{(2)} &= (1 - \alpha \; \Delta) \; X_{n-1}^{(2)} + \beta \; \Delta \; X_n^{(1)} \\ X_n^{(3)} &= X_{n-1}^{(3)} + \Delta \; X_n^{(2)} \end{cases}$$

Dynamic environment demand online tracking of obstacle in the real world. The techniques should require to model nonlinear and non-Gaussian signal to accurately draw the physical world [2]. The data arrive the system should be process online for using the storage capacity efficiently. In tracking problem, the state of object to be tracked in physical world evolve in time, a state vector has the necessary information. This information comes

from the object that .in order to minimize error, a measurement model is made, in addition to system model. The particle filter estimates the posterior probability given the observed data. The Hidden Markov Model(HMM) depicts the system in presence of hidden and observed variable. The nature of dynamical system evolves with time and predictor and update step in HMM are probabilistic.

B.2. Kalman filter for object tracking:

Kalman filter and its variants are good at modeling Gaussian distribution, i.e. only for linear model. The motion of object is linear.

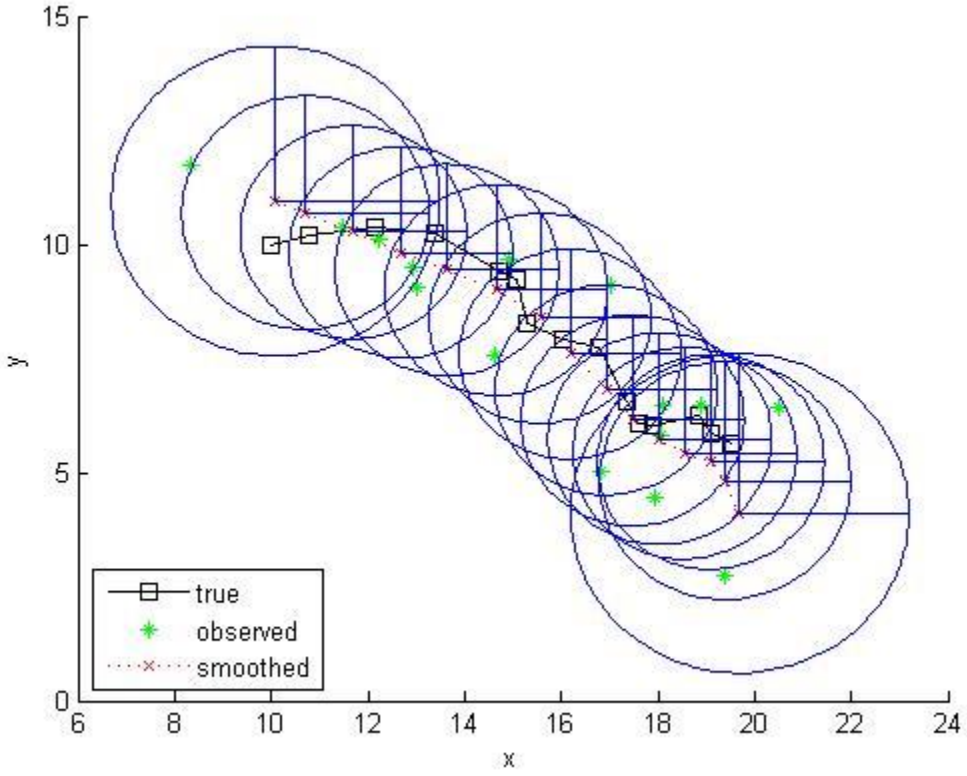
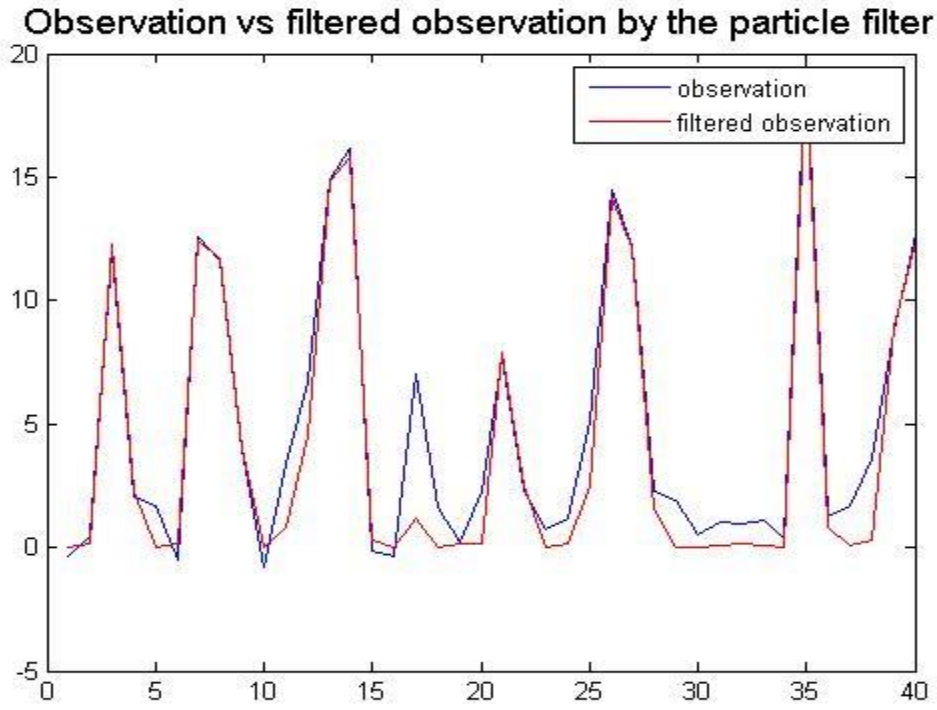


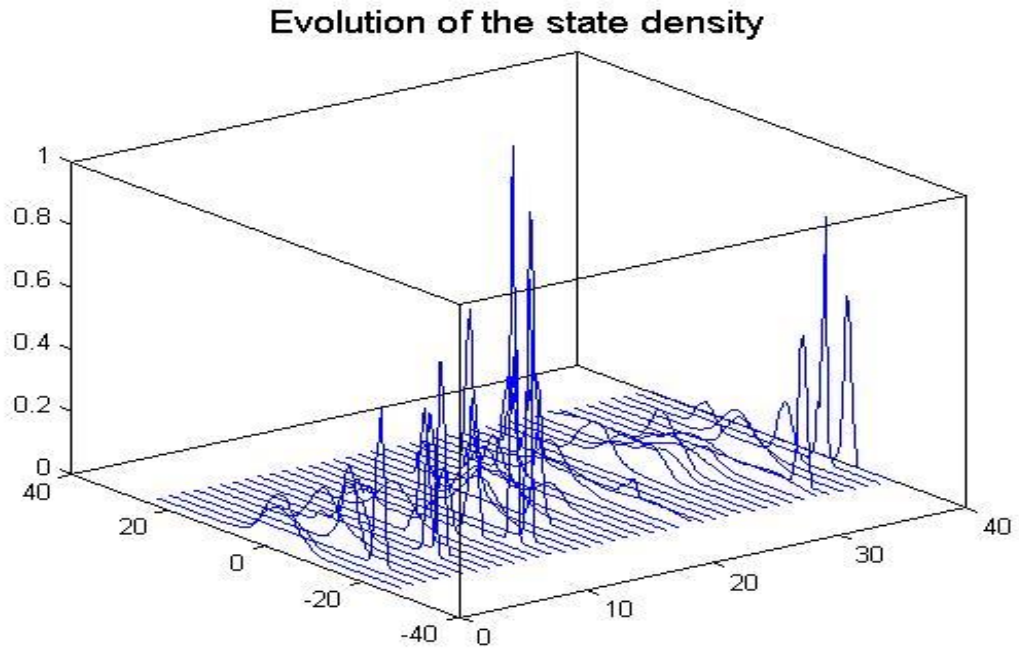
Fig 5. Tracking an object by Kalman filtering

C. Simulation Results for particle filtering:

The result in figure below are data observed by and estimation of data by particle filtering. The filter observation follows the path of observed data. The evolution of state of position of particle the peak is high probable state position.

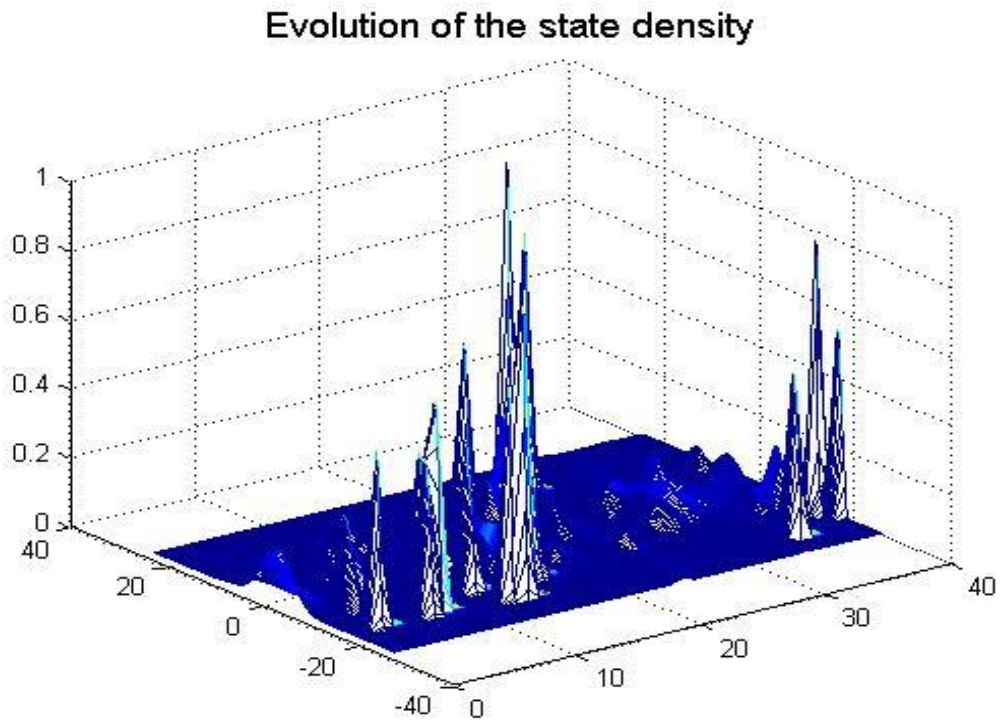


(a)

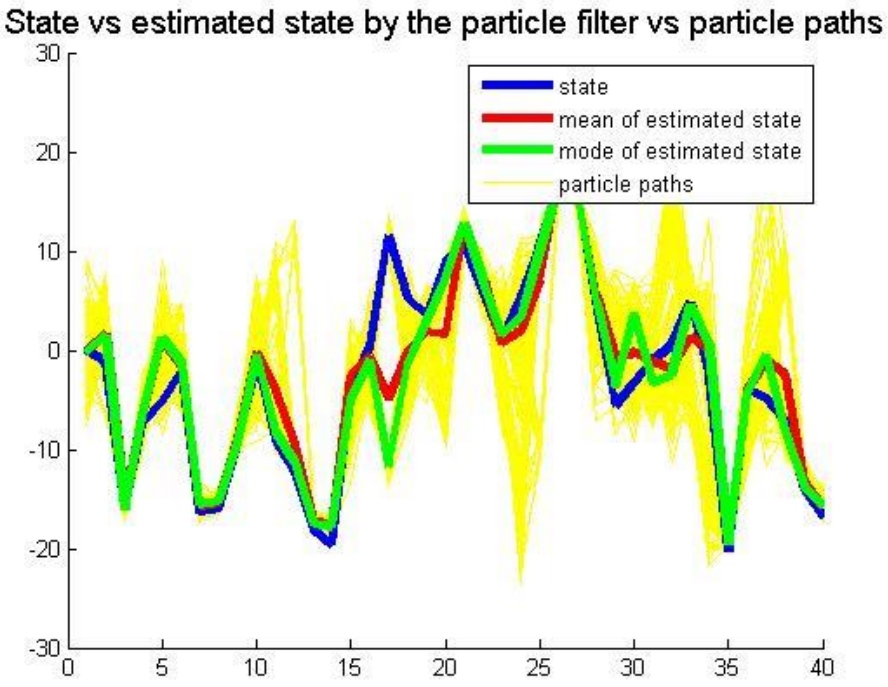


(b)

Fig 5 : a, b estimated observation and evolution of state in Particle filtering



(c)



(d)

Fig 5. (c), (d) Estimated state in Particle filtering

C.2. Simulation for tracking a red object:

Before any red object appear in the frame. The particle filter would take initial guess in the whole scene.

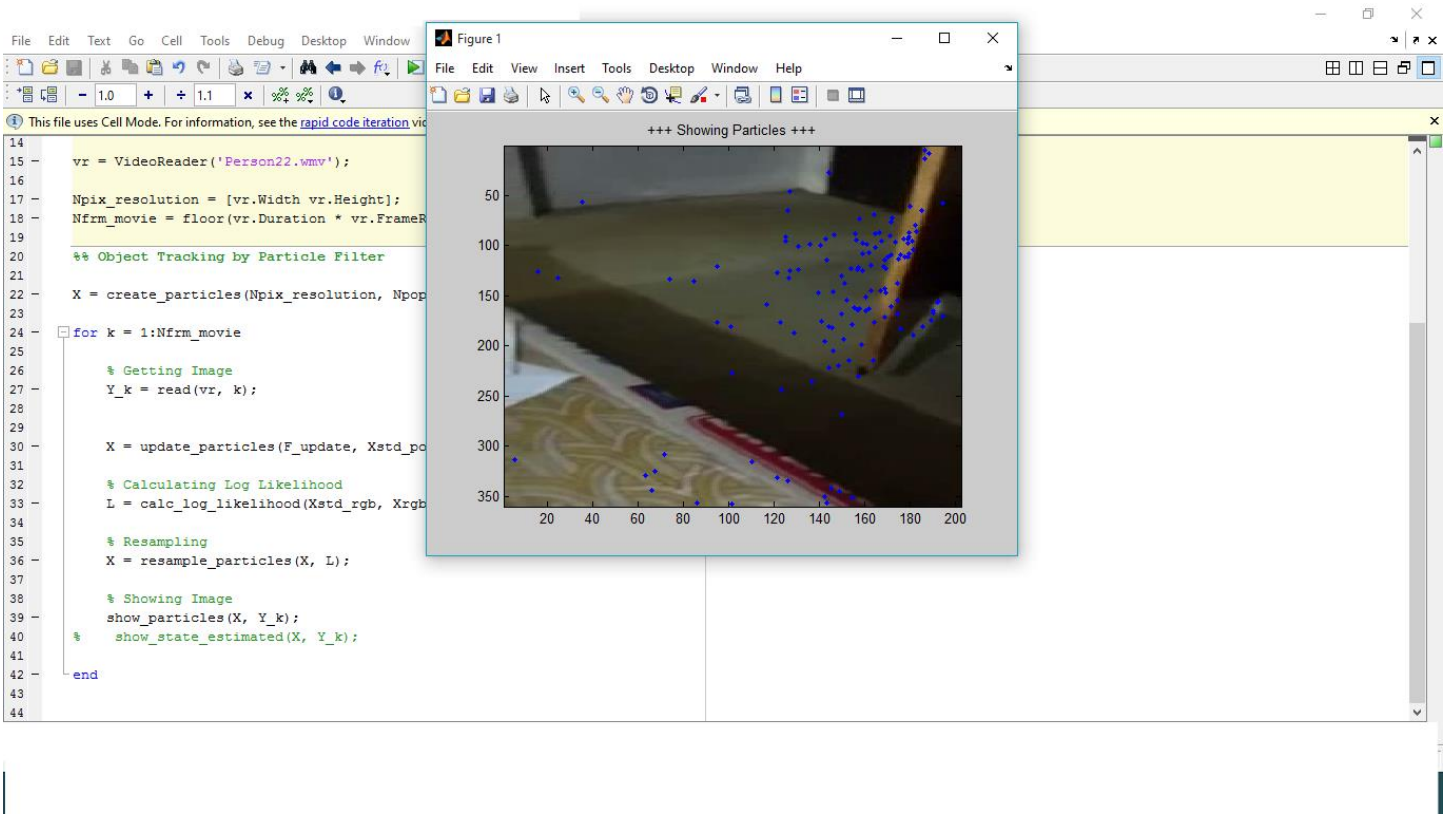


Fig 6. tracking of red box

In this simulation particle filtering is implemented for tracking red object. The code is written in Matlab. four .m files are made to run the simulation.one .m file is written for calculating likelihood of a given state. Second .m file for resampling. Third one for updating the particle states. And fourth one for displaying the particles.one main .m file that takes all value and run the simulation. The folder of the code has video file(.wmv) which has red object to be tracked.

In the first frame the scene has no object. The particles are calculating the likelihood of red object, resampling its states.

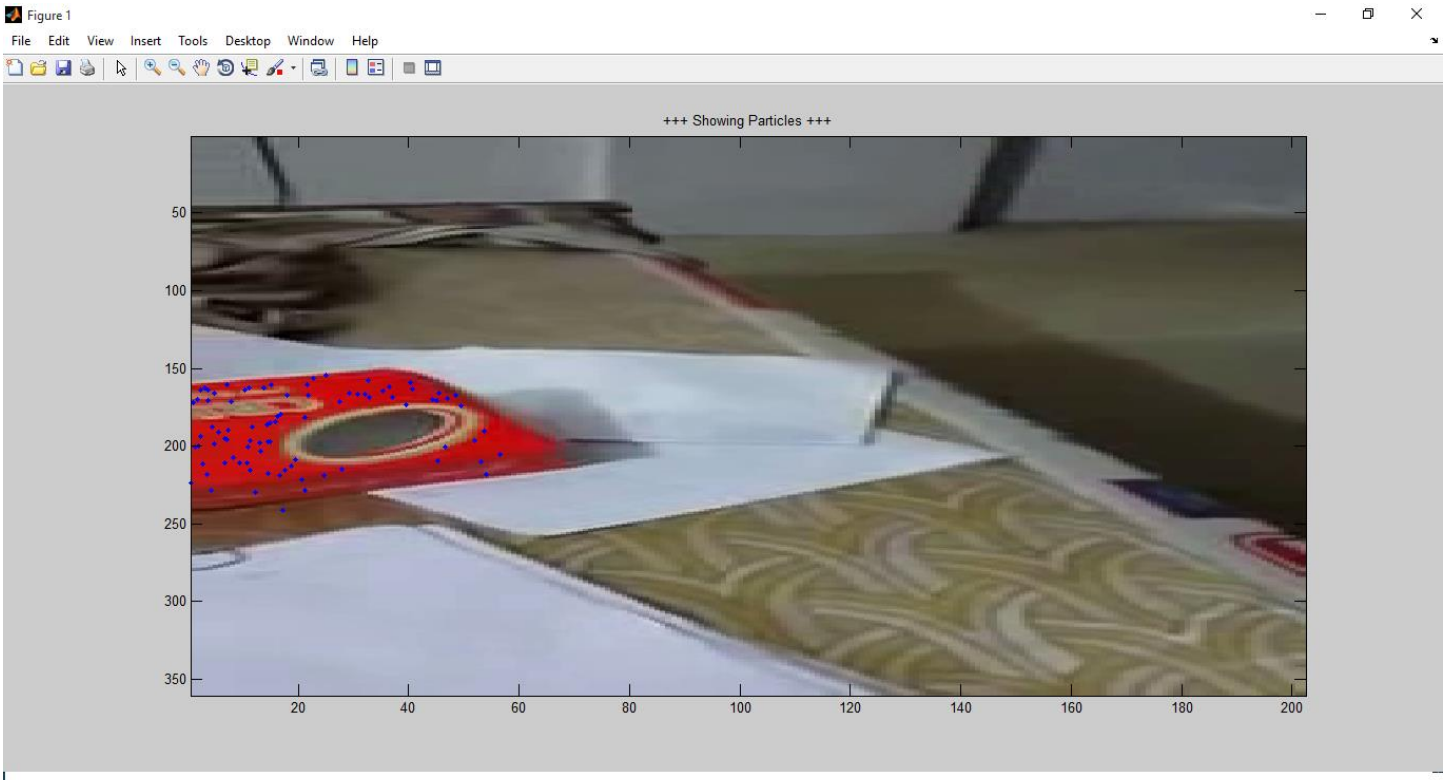


Fig 7. particles tracking the red box, appear half in this frame

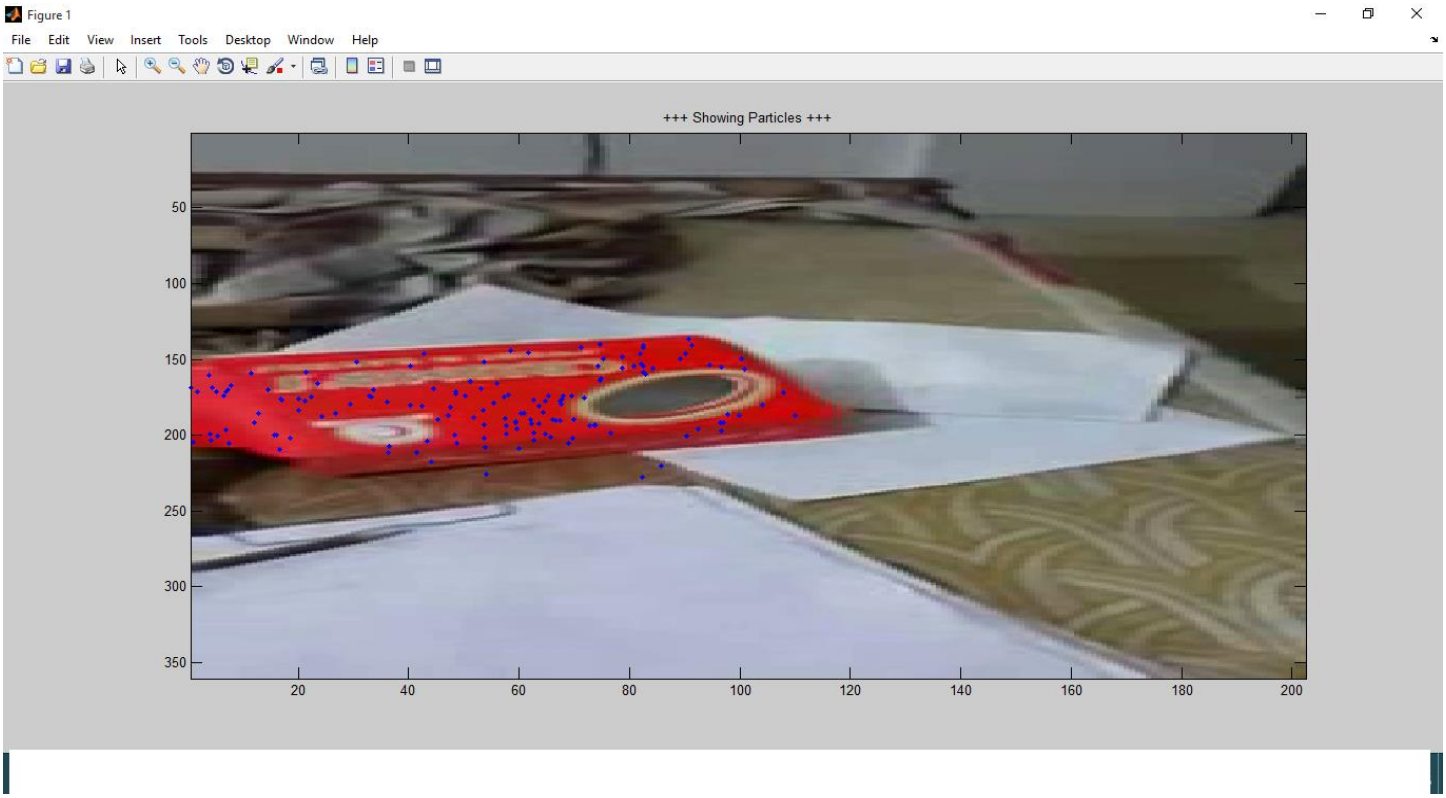
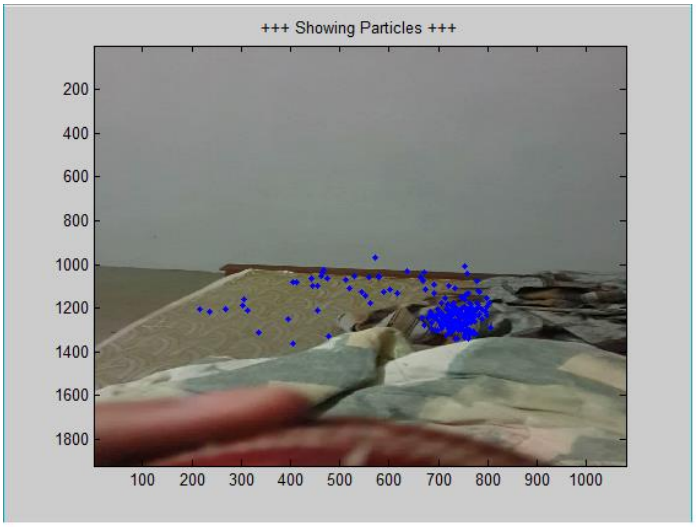
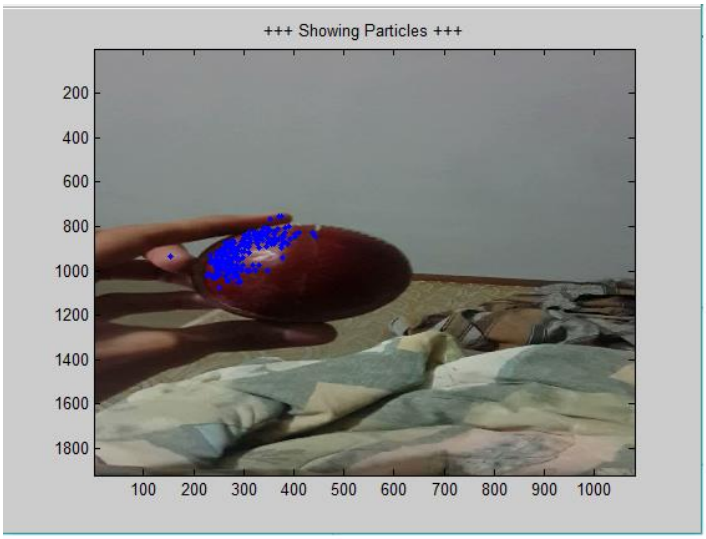


Fig 8. particles tracking the red box, appear full in this frame

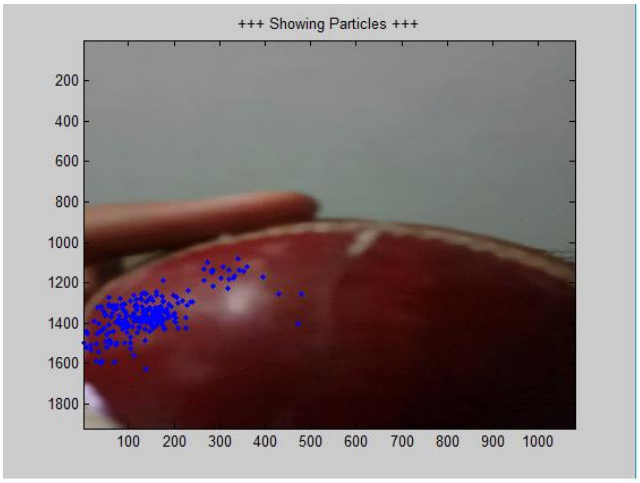
Initially the Particle are clustered around some red object which is noise for our object, but in next few frames the ball is tracked by the particles. As the ball moves in next few frames, the right side of the object gets dark, because the light is coming from the left side. So in the rest of frame, the particles are clustered to left of ball as it has red pixels.



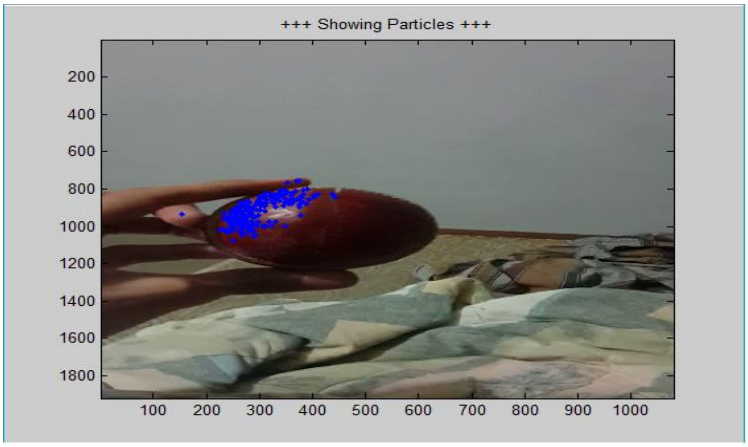
(a)



(b)



(c)



(d)

Fig 9. (a), (b), (c), and (d) Tracking of red object in different frames.

D. Coding procedure for particle filtering in Matlab:

The coding of particle filtering was done in following steps:

I. Calculate the likelihood function of chromatic pixel (R, G, B), one want to track in video
II. Generating particles, randomly in first frame, but should be spread uniformly in the frame
III. Main file which call all these sub .m files
IV. Resampling the particles. Different approaches can be done to achieve it.
V. Lastly, Display all the particles.

The video file in which one want to track object is in .wmv file, with all the other files in a folder mention above.

E. Summary

In this project report, Kalman and particle filtering has implemented for localization and object tracking. The need for Particle filtering happened when Kalman filtering and its variants could not model arbitrary (nonlinear) distribution. the simulation has been implemented for Kalman filtering in localization.in addition to it, Kalman filter has been implemented for tracking an object having Gaussian distribution. The particle filtering is used to track red object. The simulation has run in Matlab. Already a comprehensive work has been done on objet tracking, how to track multiple object in a scene. In addition to that we would add further suggestion to how to track multiple object. If frames are divided in two half plane and particle are assigned to specific half plane, so that particle don't trespass the other half plane. One can track multiple objects. The distribution of multiple object has multiple nonlinear distribution. So distinguish them is a challenging task. We have studied object tracking in Stanley ADV, in which the competition was done in desert. In those dynamic environment the ADV has challenge of its laser and camera getting dirt and dust from the zone. So Noise elimination is an essential step.

.

REFERENCES:

[1]. **:** Nicholas Metropolis, S. Ulam," The Monte- Carlo method", journal of American Statistical association, Vol.44

[2]. Arulampalam et. al. (2002). A tutorial on particle filters for online nonlinear/non-Gaussian Bayesian tracking. IEEE Transactions on Signal Processing. 50 (2). p 174—188

[3]. Thrun et al "Stanley: The robot that won the DARPA Grand Challenge"

[4]. Sabastian thrun, "Junior: The Stanford Entry in the Urban Challenge"

[5]. Fredrik Gustafsson, "Particle filtering for Positioning, Navigation and Tracking"