Shreyas Prabhakar
12th March 2018

AUE 8930: AUTONOMOUS DRIVING TECHNOLOGIES

HOMEWORK 3

Problem 1

(1) Describe the problem of SLAM and explain the difficulties.
The main problem or objective of SLAM is to explore an unknown environment. Given the robot's control and the observations made the robot of the nearby features (or landmarks), the SLAM problem is to estimate the location of the robot and the map of features (i.e., the position of landmarks).

SLAM appears to be a chicken-or-egg problem where a map is need for localizing a robot and a good pose estimate is needed to build a map. In the real world, the mapping between observations and landmarks is unknown. Picking wrong data associations can have catastrophic consequences and this error correlates data associations. Thus, SLAM is regarded as a hard problem.

(2) Briefly describe the ideas of EKF SLAM and FastSLAM. Compare their differences in terms of techniques and application cases.
The most realistic robotic problems involve nonlinear functions. These non-linear functions lead to non-gaussian distribution. Therefore, the Kalman filter will be no longer applicable.

$$x_t = g(u_t, x_{t-1})$$
$$z_t = h(x_t)$$

The equations for x_t and z_t are non-linear
In EKF SLAM we use Taylor's series expansion to linearize this non-linear function as an approximation. And then we insert this to Kalman filter equations. So, the equations become,
In scalar form,

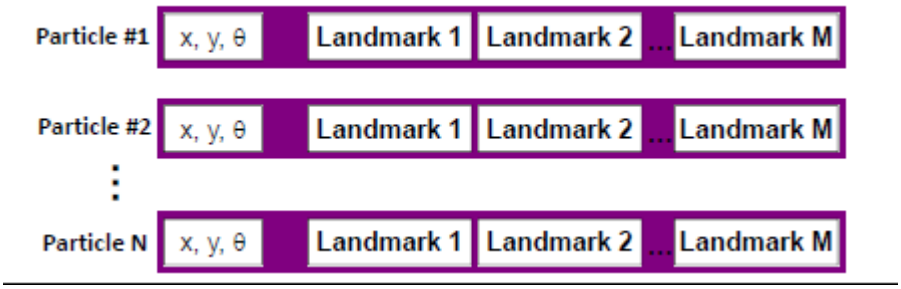
$$g(u_t, x_{t-1}) \approx g(u_t, u_{t-1}) + \frac{\partial g(u_t, u_{t-1})}{\partial x_{t-1}}(x_{t-1} - \mu_{t-1})$$
$$h(x_t) \approx h(\bar{\mu}) + \frac{\partial h(\bar{\mu})}{\partial x_t}(x_t - \bar{\mu}_t)$$

In matrix form,

$$g(u_t, x_{t-1}) \approx g(u_t, u_{t-1}) + G_t(x_{t-1} - \mu_{t-1})$$
$$h(x_t) \approx h(\bar{\mu_t}) + G_t(x_t - \bar{\mu_t})$$

The major drawbacks of EKF SLAM are, it employs linearized models of nonlinear motion and observation models and so it inherits many caveats. Also, the computational effort required grows dramatically with the number of landmarks.

To overcome these drawbacks we use Fast SLAM. The Fast SLAM uses particle filter. The particle filter represents nonlinear process model and non-Gaussian pose distribution for the robot pose estimation. It represents belief by random samples. Particle Filter uses Sampling Importace Resampling (SIR) principle.
The FastSLAM uses Rao-Blackwellized particle filtering based on the landmarks. Each landmark is represented by a 2x2 Extended Kalman Filter (EKF). Therefore each of the particle has to maintain M EKFs.

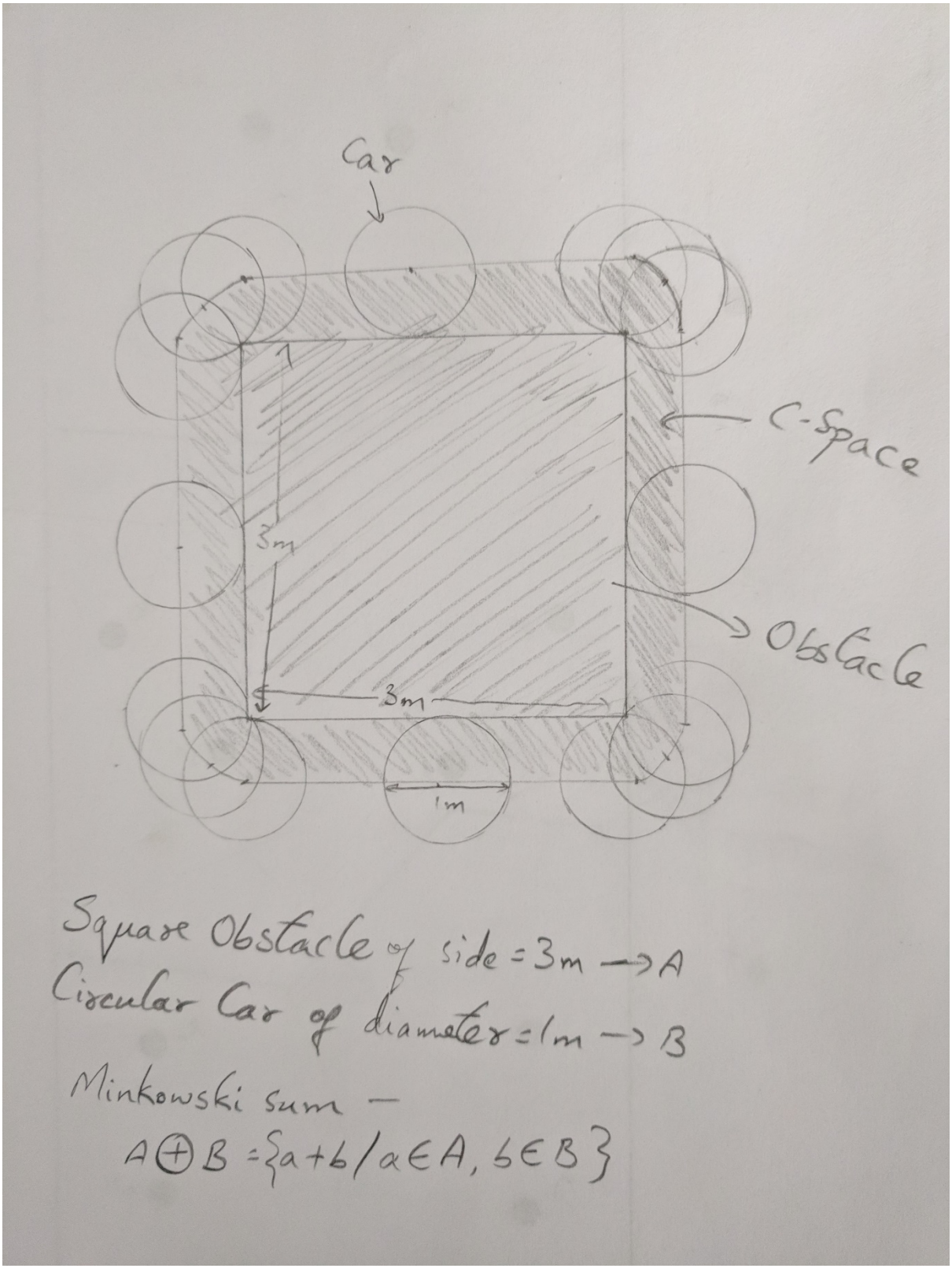


Differences between EKF SLAM and FastSLAM –

1. EKF SLAM and FastSLAM solve the same problem while making use of the identical probabilistic motion and measurement models.
2. EKF SLAM apllies Kalman Filter once to a high dimensional filtering problem where as the FastSLAM employs M*k tiny EKFs (k EKFs for each of the M particles).
3. The complexity of FastSLAM thends to be far better than the complexity of EKF SLAM.
4. FastSLAM has been tested in physical environments with as many as 50,000 landmarks, which were mapped accurately with as few as 250 particles. If we had used a EKF SLAM to keep track of the same number of features, it would need to make more than 2.5 billion covariance updates given even a single observation of a single landmark!

(3) What is C-obstacle space? Draw the C-obstacle space of a square obstacle (side length = 3 m) for a circle vehicle (diameter = 1 m) using Minkowski sum.

Configuration (C) Space of an obstacle is the space of all possible configurations of that obstacle.



(4) Briefly (2 sentences each) describe the ideas of the following motion planning approaches: roadmap, cell decomposition, potential fields, and bug algorithms.

Roadmap –

The main idea of Roadmap approach is the reduce N-dimensional configuration space (C-space) to a set of one-dimensional paths to search.

There are 2 main techniques – Visibility Graph and Voronoi Diagram.

Cell Decomposition –

The main idea of Cell Decomposition is to account for all of the free space. There are 2 main techniques – Trapezoidal Decomposition and Quadtree Decomposition.

Potential Fields –

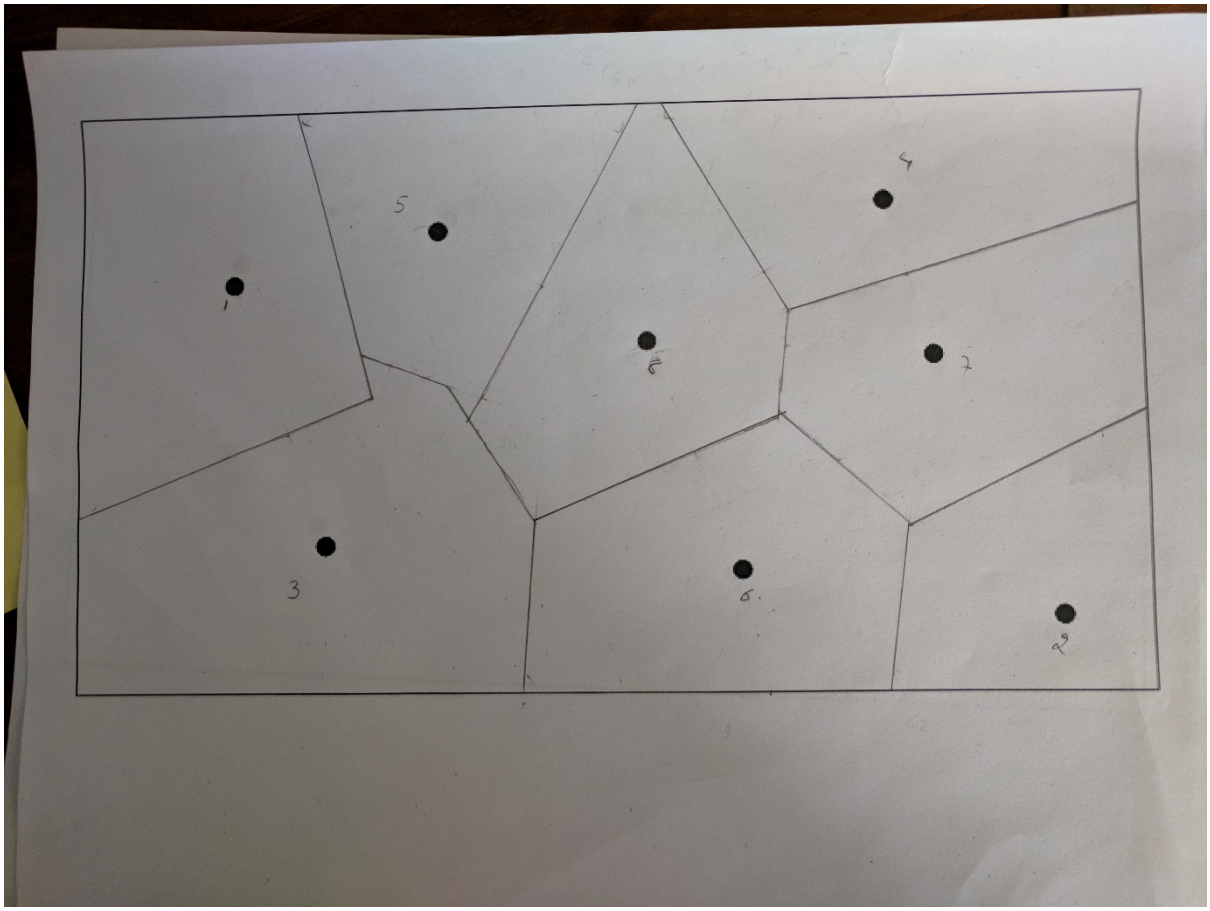
The main idea of Potential Fields is to create local control strategies, that will be more flexible than Roadmap and Cell Decomposition.

The goal location generates an attractive potential and the obstacle generates a repulsive potential. The negative gradient of the total potential is treated as an artificial force applied to the robot.

Bug algorithms –

It is used when we have limited knowledge of the map for path planning. These algorithms are inspired by the insects. There are 2 main strategies – Bug 1 and Bug 2 algorithm strategies .

(5) Draw the Voronoi diagram using L1 metric.



Problem 2: EKF SLAM

A vehicle initially stops in an unknown environment. It detects two obstacles based on its onboard lidar. The initial vector in EKF SLAM is defined as

$$\mu_0 = [x, y, \theta, x_1, y_1, x_2, y_2\ ]^T = [0, 0, 0, 147, 102, 98, 53\ ]^T$$

where (x, y, θ) is the robot states, and (x_1, y_1) and (x_2, y_2) are locations of the two objects.

The vehicle moves in a straight line with a constant speed $v_t = 1\ m/s$, ($\omega_t = 0$). As the vehicle moves, the lidar detects the relative distances and relative angles of the two objects: (r^1, ϕ^1) and (r^2, ϕ^2) in units of meter and radian, which are saved in file "s1.mat" and "s2.mat". The sampling time is $\Delta t = 1\ s$.

The noises in the motion and sensing are both Gaussian $N(0, R)$ and $N(0, Q)$, where

$$R = \begin{bmatrix} 0.1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0.1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0.1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \quad Q = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0.1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0.01 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0.1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0.01 \end{bmatrix}$$

Implement EKF SLAM in Matlab to find out the locations of the robot and two objects, i.e., update μ based on the data in "s1.mat" and "s2.mat".

Given –

$$\mu_0 = [x, y, \theta, x_1, y_1, x_2, y_2\ ]^T = [0, 0, 0, 147, 102, 98, 53\ ]^T$$

$v_t = 1\ m/s$,
($\omega_t = 0$).

$$\Delta t = 1\ s$$

(r^1, ϕ^1) and (r^2, ϕ^2) for landmark 1 and landmark 2 respectivley.

Noises in motion and sensing,

$$R = \begin{bmatrix} 0.1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0.1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0.1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \quad Q = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0.1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0.01 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0.1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0.01 \end{bmatrix}$$

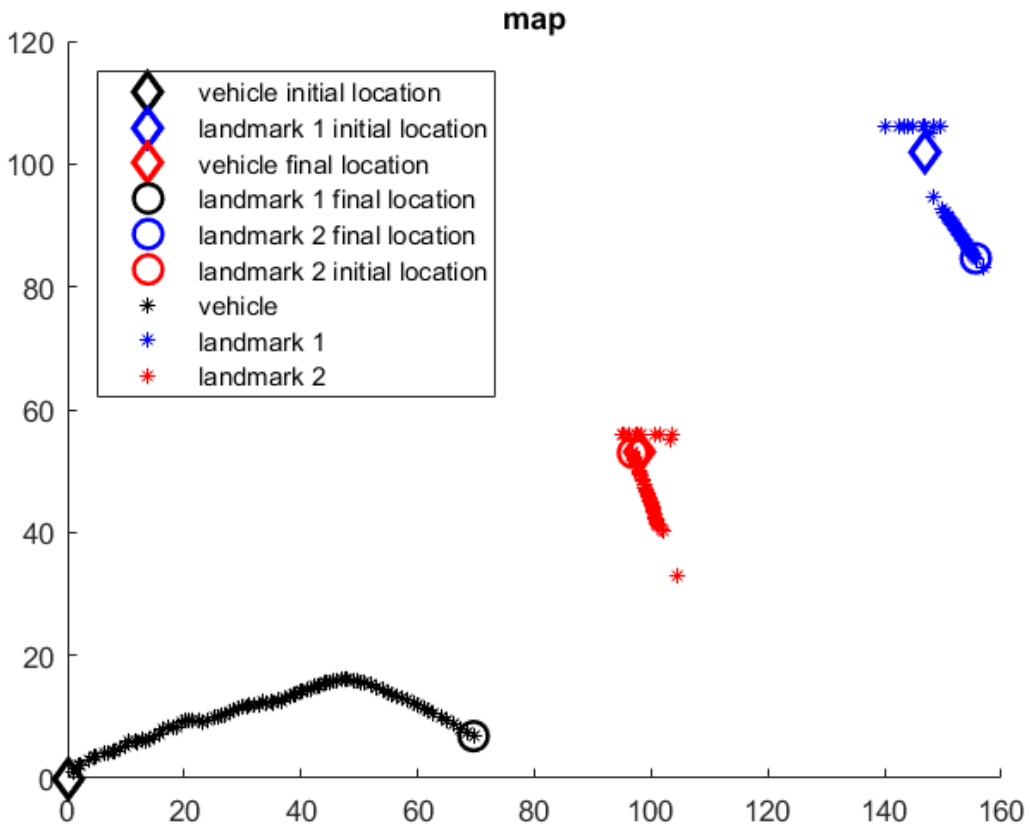
Assumptions and approximations made –

$$\sum_0 = \begin{matrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 10^{150} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 10^{150} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 10^{150} & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 10^{150} \end{matrix}$$

Ideally the diagonal elements corresponding to the diagonal elements are supposed to be infinite(or as large as possible) for the initial assumption, I have considered it to be equal to 10^{150}.

And in the question it is given that wt = 0. But since wt is in the denominator, the term containing wt becomes indeterminant. So I have assumed wt to have some tolerance value epsilon $\in = 10^{-150}$

The location of the vehicle, landmark 1 and landmark 2 after each iteration/observation is as shown in figure below.



The final **μ** value is

$$\mu_{100} = [x, y, \theta, x_1, y_1, x_2, y_2\ ]^T$$
$$= [\ \ 69.5606, 6.8339, -0.4320, 155.5864, 84.6311, 96.8638, 52.9673\ ]^T$$

Matlab code is attached at hw3.m

The μ values after every iteration / observation is in the mu.txt file.

Matlab code –

```
close all
clear all
load s1.mat
load s2.mat

s{1} = s1;
s{2} = s2;
clear s1 s2

mu0 = [0, 0, 0, 147, 102, 98, 53]';
lengthMu = size(mu0,1);
numLandmarks = (lengthMu - 3)/2;
sigma0 = zeros(lengthMu);
for i = (lengthMu - (numLandmarks*2) + 1):7
sigma0(i,i) = 10^150;
end

vt = 1; %velocity in  m/s
wt = 10^-150;%radial velocity (straight line)
dt = 1;%sampling time in s

R = zeros(lengthMu);
for i=1:3
R(i,i) = 0.1;
end
Q = zeros(lengthMu);
Q((lengthMu - (numLandmarks*2) + 1): lengthMu, (lengthMu -
(numLandmarks*2)       +       1):       lengthMu)       =
diag([0.1,0.01,0.1,0.01]);

for i = 1:size(s{1},2)
Qt{i} = Q(3+((i-1)*2+(1:2)),3+((i-1)*2+(1:2)));
end

mut1 = mu0;
sigmat1 = sigma0;
Fx = [eye(3),zeros(3,numLandmarks*2)];
vw = vt/wt;
wdt = wt*dt;


for i = 1:size(s{1},1)
%%
%Prediction
```

```
temp = [(-vw*sin(mut1(3)))+(vw*sin(mut1(3)+wdt));
(-vw*sin(mut1(3)))+(vw*sin(mut1(3)+wdt));
wdt];
mut = mut1 + Fx' * temp;
temp(3,1) = 0;
temp = [zeros(3,2), temp];
Gt = eye(lengthMu) + Fx' * temp * Fx;
sigmat = (Gt*sigmat1*Gt') + R;

%%
%Correction
for j = 1:numLandmarks
delta = [mut(3+((j-1)*2+(1:2))) - mut(1:2)];
q = delta' * delta;
zt = [sqrt(q); atan2(delta(2),delta(1))-mut(3)];
Fxj         =         [Fx;         zeros(2,3),zeros(2,2*j-
2),eye(2),zeros(2,2*size(s{1},2)-2*j)];
Ht  =  (1/q  .*  [-sqrt(q)*delta(1),  -sqrt(q)*delta(2),  0,
sqrt(q)*delta(1), sqrt(q)*delta(2); delta(2), -delta(1), -
q, -delta(2), delta(1)]) * Fxj;
Kt    =    sigmat*Ht'*pinv(Ht*sigmat*Ht'    +    Q(3+((j-
1)*2+(1:2)),3+((j-1)*2+(1:2))));

mut = mut + (Kt*(s{j}(i,:)' - zt));
sigmat = (eye(7) - Kt*Ht)*sigmat;
clear delta q zt Fxj Ht Kt
end

mut1 = mut;
sigmat1 = sigmat;
mu{i} = mut;
muu(i,:) = mut;
sigma{i} = sigmat;
clear mut sigmat


end

figure;
hold on

plot(mu0(1), mu0(2), 'kd', 'LineWidth',2,'MarkerSize',10);
plot(mu0(4), mu0(5), 'bd', 'LineWidth',2,'MarkerSize',10);
plot(mu0(6), mu0(7), 'rd', 'LineWidth',2,'MarkerSize',10);

plot(muu(100,1),             muu(100,2),             'ko',
'LineWidth',1.5,'MarkerSize',10);
```

```
plot(muu(100,4),              muu(100,5),              'bo',
'LineWidth',1.5,'MarkerSize',10);
plot(muu(100,6),              muu(100,7),              'ro',
'LineWidth',1.5,'MarkerSize',10);

for i = 1:size(muu,1)
plot(muu(i,1), muu(i,2), 'k*','MarkerSize',5);
plot(muu(i,4), muu(i,5), 'b*','MarkerSize',5);
plot(muu(i,6), muu(i,7), 'r*','MarkerSize',5);
end
hold off
title("map");
legend("vehicle initial location", "landmark 1 initial
location","vehicle final location", "landmark 1 final
location", "landmark 2 final location", "landmark 2 initial
location",    "vehicle",    "landmark    1",    "landmark
2",'Location','northwest');

clear mu
mu = muu;
T = table(mu);
writetable(T,'mu.txt','Delimiter', ' ')
```