Ben Gurion University of the Negev

Faculty of Engineering Sciences

Intelligent Robotics System

Exercise 2

Monte Carlo Localization of a mobile robot using landmarks

Nir Dgani

Lecturer: Dr. Armin Biess

Date: 06/06/2017

Monte Carlo localization of a mobile robot using landmarks

The robot has three degrees of freedom (DOFs):

$$\boldsymbol{x} = \{x, y, \theta\}$$

The world size is: 100 x 100

The world includes four landmarks at:

$$\boldsymbol{m} = \begin{Bmatrix} m_1 \\ m_2 \\ m_3 \\ m_4 \end{Bmatrix} = \begin{Bmatrix} \{20, 20\} \\ \{80, 80\} \\ \{20, 80\} \\ \{80, 20\} \end{Bmatrix}$$

The robot can take two motor commands, u_1 and u_2:

- u_1 is a forward movement command. ($u_1 > 0$).

$$\boldsymbol{u_1} = u_1 \cos\theta' \,\hat{\boldsymbol{i}} + u_1 \sin\theta' \,\hat{\boldsymbol{j}}$$

- u_2 is a turn movement command. ($0 \leq u_2 < 2\pi$).

The deterministic motion model of the robot is given by:

$$\theta' = \theta + u_2$$

$$x' = x + u_1 \cos\theta'$$

$$y' = y + u_1 \sin\theta'$$

a. Three poses for robots a, b and c:

$$\boldsymbol{x_a} = \{40,40,0\}$$

$$\boldsymbol{x_b} = \left\{60,50,\frac{\pi}{2}\right\}$$

$$\boldsymbol{x_c} = \left\{30,70,\frac{3\pi}{4}\right\}$$

The Matlab code:

```
% Set the Robots
myrobot_a = cRobot();
myrobot_b = cRobot();
myrobot_c = cRobot();
myrobot_a.set(40,40,0);
myrobot_b.set(60,50,pi/2);
myrobot_c.set(30,70,3*pi/4);
Robot_style='robot';
myworld = cWorld();
myworld.plot;
hold on;
myrobot_a.plot('blue',Robot_style);
myrobot_b.plot('red',Robot_style);
myrobot_c.plot('green',Robot_style);

% prints the poses of the robots 'a' 'b' and 'c' to the
Matlab prompt
disp('robot a:');
myrobot_a.print;
disp('robot b:');
myrobot_b.print;
disp('robot b:');
myrobot_c.print;
```

Matlab plot:

```
>> Ex2_Dgani_IRS
robot a:
[x= 40  y=40  heading=0]
robot b:
[x= 60  y=50  heading=1.5708]
robot b:
[x= 30  y=70  heading=2.3562]
```

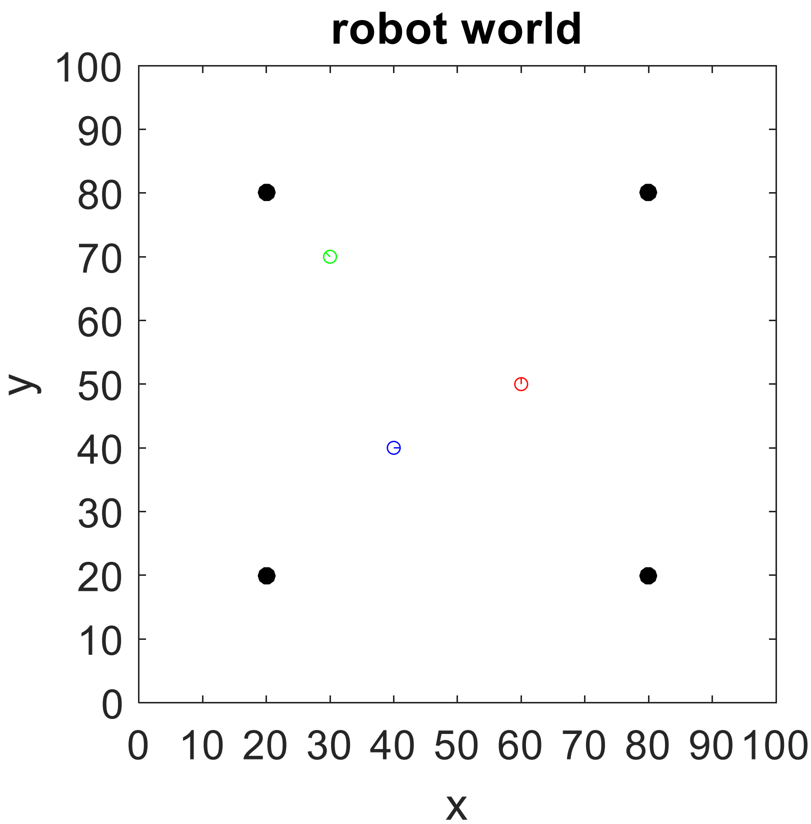

*Figure 1 – the 'world' with size 100x100. The **black** is the landmarks and the robots are displayed in **blue** (Robot a), **red** (robot b) and **green** (robot c).*

b. The function `move`.

The motor commands are noisy and corrupted with Gaussin noise:

- Forward noise: $\epsilon_F = \mathcal{N}(0, \sigma_1) = \mathcal{N}(0,5)$
- Turn noise: $\epsilon_T = \mathcal{N}(0, \sigma_2) = \mathcal{N}(0,0.1[\mathrm{rad}])$

The commends with the noise:

$$u_1{}' = u_1 + \epsilon_F = u_1 + \mathcal{N}(0, \sigma_1) = \mathcal{N}(u_1, \sigma_1)$$
$$u_2{}' = u_2 + \epsilon_T = u_2 + \mathcal{N}(0, \sigma_2) = \mathcal{N}(u_2, \sigma_2)$$

The new pose is:

$$\theta' = \theta + u_2{}'$$
$$x' = x + u_1{}' \cos \theta'$$
$$y' = y + u_1{}' \sin \theta'$$

The code of the function `move`:

```
function [ x, y, theta ] = move( obj, u_1, u_2 )
% The 'move' function takes as input then two motor commands
[u_1, u_2]
% and outputs the new pose [x, y, theta]

u_1_N = normrnd(u_1, obj.forward_noise);
u_2_N = normrnd(u_2, obj.turn_noise);

N_theta = obj.theta + u_2_N;
N_x = obj.x + u_1_N*cos(N_theta);
N_y = obj.y + u_1_N*sin(N_theta);
% theta
    if N_theta >= 2*pi
        theta = N_theta - 2*pi;
    else if N_theta < 0
            theta = N_theta + 2*pi;
        else
            theta = N_theta;
        end
    end
% x position
    if N_x >= cWorld.world_size % world_size = 100
        x = N_x - cWorld.world_size;
    else if N_x < 0
            x = N_x + cWorld.world_size;
        else
            x = N_x;
        end
    end
% y position
    if N_y >= cWorld.world_size % world_size = 100
        y = N_y - cWorld.world_size;
    else if N_y < 0
            y = N_y + cWorld.world_size;
        else
            y = N_y;
        end
    end
end
```

c. The function `sense`.

The measurements are corrupted by Gaussin noise:

$$\epsilon_S = \mathcal{N}(0, r) = \mathcal{N}(0,5)$$

```
function [SL1, SL2, SL3, SL4] = sense(obj, myworld)
% The 'sense' function outputs the distance to four
landmarks: [SL1, SL2, SL3, SL4]
% SLi: Sense distance of Landmarks 'i'
% RLi: Real distance of Landmarks 'i'

L1_x = myworld.landmarks(1,1);
L1_y = myworld.landmarks(1,2);
L2_x = myworld.landmarks(2,1);
L2_y = myworld.landmarks(2,2);
L3_x = myworld.landmarks(3,1);
L3_y = myworld.landmarks(3,2);
L4_x = myworld.landmarks(4,1);
L4_y = myworld.landmarks(4,2);

% Real Distance
RL1 = sqrt((obj.x-L1_x)^2 + (obj.y-L1_y)^2);
RL2 = sqrt((obj.x-L2_x)^2 + (obj.y-L2_y)^2);
RL3 = sqrt((obj.x-L3_x)^2 + (obj.y-L3_y)^2);
RL4 = sqrt((obj.x-L4_x)^2 + (obj.y-L4_y)^2);

% Distance + noise // sense_distance_noise = N(0,r)
SL1 = normrnd(RL1, obj.sense_distance_noise);
SL2 = normrnd(RL2, obj.sense_distance_noise);
SL3 = normrnd(RL3, obj.sense_distance_noise);
SL4 = normrnd(RL4, obj.sense_distance_noise);
end
```

d. The function `measurement_probability`.

The probability of landmark 'i' is:

$$P_i = \frac{1}{\sqrt{2\pi\sigma_N^2}} \exp\left(-\frac{(RL_i - SL_i)^2}{(2\sigma_N^2)}\right)$$

Where:

- RL_i is the real distance to landmark 'i'.
- SL_i is the measured distance to landmark 'i'.
- σ_N is the sense distance noise.

```
function MP = measurement_probability(obj, myworld, SL1,
SL2, SL3, SL4 )
% The 'measurement_Probability' function takes the robot's
measurments as
% input and outputs the probability (MP) for this measurment
to be observed
% when being in pose x.
% PLi: Probability of Landmarks i

L1_x = myworld.landmarks(1,1);
L1_y = myworld.landmarks(1,2);
L2_x = myworld.landmarks(2,1);
L2_y = myworld.landmarks(2,2);
L3_x = myworld.landmarks(3,1);
L3_y = myworld.landmarks(3,2);
L4_x = myworld.landmarks(4,1);
L4_y = myworld.landmarks(4,2);

% Real Distance
RL1 = sqrt((obj.x-L1_x)^2 + (obj.y-L1_y)^2);
RL2 = sqrt((obj.x-L2_x)^2 + (obj.y-L2_y)^2);
RL3 = sqrt((obj.x-L3_x)^2 + (obj.y-L3_y)^2);
RL4 = sqrt((obj.x-L4_x)^2 + (obj.y-L4_y)^2);

DN = obj.sense_distance_noise; % Distance noise
PL1 = 1/sqrt((2*pi)*DN^2)*exp(-((RL1-SL1)^2)/(2*SL1)^2);
PL2 = 1/sqrt((2*pi)*DN^2)*exp(-((RL2-SL2)^2)/(2*SL2)^2);
PL3 = 1/sqrt((2*pi)*DN^2)*exp(-((RL3-SL3)^2)/(2*SL3)^2);
PL4 = 1/sqrt((2*pi)*DN^2)*exp(-((RL4-SL4)^2)/(2*SL4)^2);

MP = PL1 * PL2 * PL3 * PL4; % The Total probability
end
```

e. The initial pose of the robot is:

$$\boldsymbol{x}_{initial} = \begin{Bmatrix} x_0 \\ y_0 \\ \theta_0 \end{Bmatrix} = \begin{Bmatrix} 10 \\ 15 \\ 0 \end{Bmatrix}$$

The robot performs the following five actions:

1. $u_{11} = 60,\ u_{21} = 0$
2. $u_{12} = 30,\ u_{22} = \frac{\pi}{3}$
3. $u_{13} = 30,\ u_{23} = \frac{\pi}{4}$
4. $u_{14} = 20,\ u_{24} = \frac{\pi}{4}$
5. $u_{15} = 40,\ u_{25} = \frac{\pi}{4}$

The robots poses:

$$\boldsymbol{x}_i = \begin{Bmatrix} x_i \\ y_i \\ \theta_i \end{Bmatrix} = \begin{Bmatrix} x_{i-1} + u_{1i} \cos \theta_i \\ y_{i-1} + u_{1i} \sin \theta_i \\ \theta_{i-1} + u_{2i} \end{Bmatrix}$$

$$\boldsymbol{x}_1 = \begin{Bmatrix} x_1 \\ y_1 \\ \theta_1 \end{Bmatrix} = \begin{Bmatrix} x_0 + u_{11} \cos 0 \\ y_0 + u_{11} \sin 0 \\ \theta_0 + u_{21} \end{Bmatrix} = \begin{Bmatrix} 10 + 60 \\ 15 + 0 \\ 0 + 0 \end{Bmatrix} = \begin{Bmatrix} 70 \\ 15 \\ 0 \end{Bmatrix}$$

$$\boldsymbol{x}_2 = \begin{Bmatrix} x_2 \\ y_2 \\ \theta_2 \end{Bmatrix} = \begin{Bmatrix} 70 + 30 \cos\left(\frac{\pi}{3}\right) \\ 15 + 30 \sin\left(\frac{\pi}{3}\right) \\ 0 + \left(\frac{\pi}{3}\right) \end{Bmatrix} = \begin{Bmatrix} 85 \\ 15\left(1 + \sqrt{3}\right) \\ \left(\frac{\pi}{3}\right) \end{Bmatrix} = \begin{Bmatrix} 85 \\ 40.981 \\ 1.047 \end{Bmatrix}$$

$$\boldsymbol{x}_3 = \begin{Bmatrix} x_3 \\ y_3 \\ \theta_3 \end{Bmatrix} = \begin{Bmatrix} 85 + 30 \cos\left(\frac{7\pi}{12}\right) \\ 40.981 + 30 \sin\left(\frac{7\pi}{12}\right) \\ \left(\frac{\pi}{3}\right) + \left(\frac{\pi}{4}\right) \end{Bmatrix} = \begin{Bmatrix} 77.235 \\ 69.96 \\ \left(\frac{7\pi}{12}\right) \end{Bmatrix}$$

$$
\boldsymbol{x}_4 = \begin{Bmatrix} x_4 \\ y_4 \\ \theta_4 \end{Bmatrix} = \begin{Bmatrix} 77.235 + 20\cos\left(\frac{5\pi}{6}\right) \\ 69.96 + 20\sin\left(\frac{5\pi}{6}\right) \\ \left(\frac{7\pi}{12}\right) + \left(\frac{\pi}{4}\right) \end{Bmatrix} = \begin{Bmatrix} 59.915 \\ 79.96 \\ \left(\frac{5\pi}{6}\right) \end{Bmatrix}
$$

$$
\boldsymbol{x}_5 = \begin{Bmatrix} x_5 \\ y_5 \\ \theta_5 \end{Bmatrix} = \begin{Bmatrix} 59.915 + 40\cos\left(\frac{13\pi}{12}\right) \\ 79.96 + 40\sin\left(\frac{13\pi}{12}\right) \\ \left(\frac{5\pi}{6}\right) + \left(\frac{\pi}{4}\right) \end{Bmatrix} = \begin{Bmatrix} 21.278 \\ 69.607 \\ \left(\frac{13\pi}{12}\right) \end{Bmatrix}
$$

```
myworld = cWorld();
myrobot = cRobot();

% Lmotion is matrix size 5x2 for five actions : u_1i, u_2i
Lmotion = [60 30 30 20 40;
           0 pi/3 pi/4 pi/4 pi/4];

Robot5 = cRobot();

% initial pose
Robot5.set(10,15,0);
path_e(:,1) = [10,15,0];

% plots
myworld.plot; hold on;
Robot5.plot('black', 'robot');

clrS =  hsv(5);
for k = 1:5
   [xx, yy, tt] = Robot5.move(Lmotion(1,k), Lmotion(2,k));
   Robot5.set(xx, yy, tt);
   Robot5.plot(clrS(k,:), 'robot');
   path_e(:,k+1) = [xx, yy, tt];
   if k > 1
        line([path_e(1,k) path_e(1,k-1)],[path_e(2,k)
path_e(2,k-1)],'Color',clrS(k-1,:),'LineStyle',
':','LineWidth', 2);
   end
end
line([path_e(1,6) path_e(1,5)],[path_e(2,6)
path_e(2,5)],'Color',clrS(5,:),'LineStyle', ':','LineWidth',
2);
```

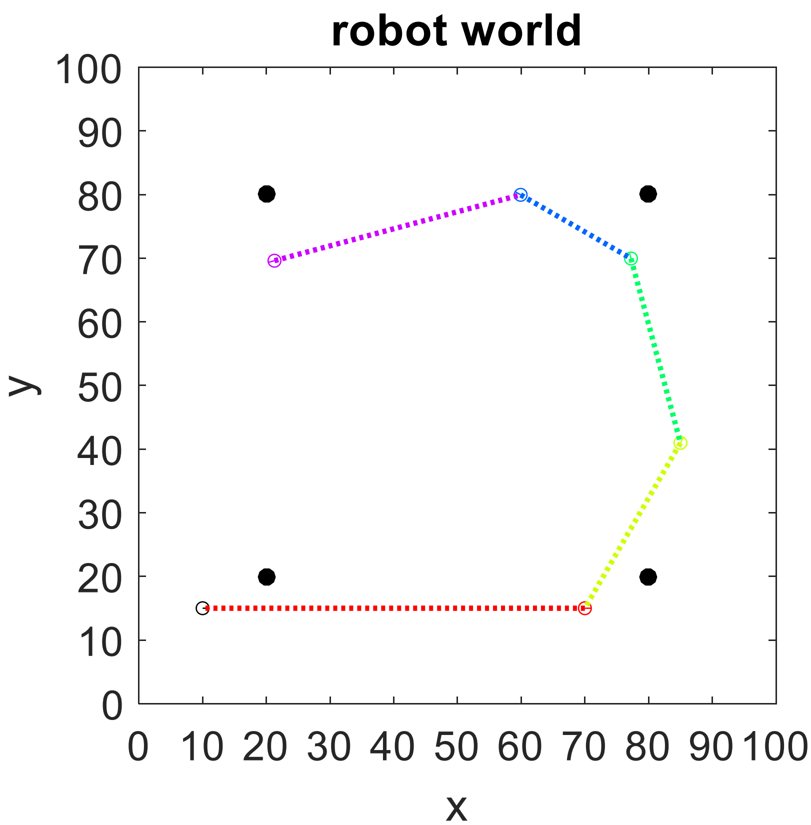

Figure 2 – the plot of five robot poses and the robot path (dotted line)

f. The true robot poses and the robot path.

Forward noise: $\epsilon_F = \mathcal{N}(0,\sigma_1) = \mathcal{N}(0{,}5)$

Turn noise: $\epsilon_T = \mathcal{N}(0,\sigma_2) = \mathcal{N}(0{,}0.1[\mathrm{rad}])$

```
% ...in continue to section 'e'

Robot5_f = cRobot();

% noises
F_noise = 5; % Forward noise
T_noise = 0.1; % Turn noise
SD_noise = 5; % Sense_distance_noise
Robot5_f.set_noise(F_noise, T_noise, SD_noise);

% initial pose
Robot5_f.set(10,15,0);
path_f(:,1) = [10,15,0];

% plots
Robot5_f.plot('black', 'robot');

clrS =  hsv(5);
for k = 1:5
   [xxf, yyf, ttf] = Robot5_f.move(Lmotion(1,k),
Lmotion(2,k));
   Robot5_f.set(xxf, yyf, ttf);
   Robot5_f.plot(clrS(k,:), 'robot');
   path_f(:,k+1) = [xxf, yyf, ttf];
   line([path_f(1,k) path_f(1,k+1)],[path_f(2,k)
path_f(2,k+1)],'Color',clrS(k,:),'LineWidth', 2);
end
```

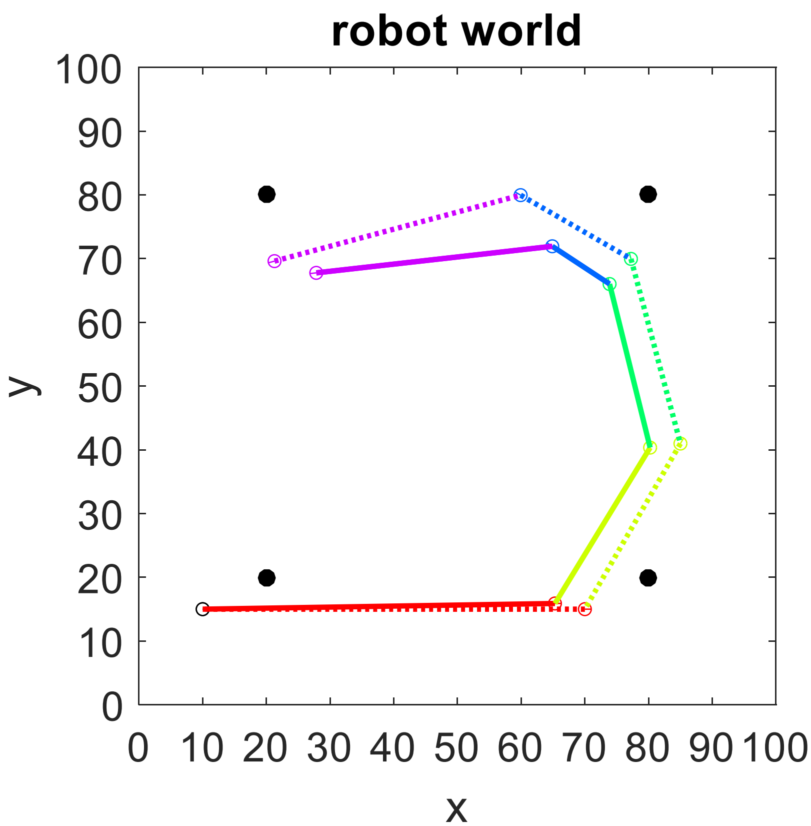

Figure 3 - The plot of true five robot poses and the robot path (as a solid line) and the five robot poses and the robot path resulting from the action command without noises (dotted line)

g. Implement a particle filter with $N = 1000$ particles to estimate the poses and the path of the robot by taking measurements of the landmarks.

The initial pose:

$$\boldsymbol{x}_{initial} = \begin{Bmatrix} x_0 \\ y_0 \\ \theta_0 \end{Bmatrix} = \begin{Bmatrix} 10 \\ 15 \\ 0 \end{Bmatrix}$$

```
% ...in continue to section ‘f’

Part_Num = 1000; % Particles Number

% initilize
path_g = zeros(3,length(Lmotion));
Bfr_res = zeros(Part_Num,3);
Bfr_res_P = zeros(Part_Num,1);
Bfr_res_W = zeros(Part_Num,1);

% initial pose and particles
Robot5_g = cRobot();
Robot5_g.set(10,15,0);
path_g(:,1) = [10,15,0];
Particle = cRobot();
Particle.set_noise(F_noise, T_noise, SD_noise);



PBC = 'black'; % The black color of particles before the
resempling
PAC = [0.5 0.5 0.5]; % The gray color of particles after the
resempling
RC = 'cyan'; % The cyan color of the robot

% plot
Robot5_g.plot(RC, 'robot');

for k = 1:5
    % Measure the distance from the true pose
    Robot5_f.set(path_f(1,k+1), path_f(2,k+1),
path_f(3,k+1));
    [MDL1, MDL2, MDL3, MDL4] =  Robot5_f.sense( myworld);
    % MDLi: the measured distance of landmark 'i'
```

```
    % Prediction Step
    for j = 1:Part_Num
        Particle.set(path_g(1,k), path_g(2,k), path_g(3,k));
        [xxg, yyg, ttg] = Particle.move(Lmotion(1,k),
Lmotion(2,k));
        Particle.set(xxg, yyg, ttg);
        Particle.plot(PBC, 'particle');
        Bfr_res(j,:) = [xxg, yyg, ttg]; % Pose before the
resempling [1000x3]
        Bfr_res_P(j) =
Particle.measurement_probability(myworld, MDL1, MDL2, MDL3,
MDL4); % Probability before the resempling [1000x1]
    end
    P_sum = sum(Bfr_res_P);
    for i = 1:Part_Num
        Bfr_res_W(i) = Bfr_res_P(i)/P_sum; % Weight
    end

    % Resempling
    L = length(Bfr_res_W);
    rnd = rand*L^(-1);
    W1 = Bfr_res_W(1);
    a = 1; b = 1;
    Aft_res = zeros(1000,3); % Initilize the pose after the
resempling
        for q = 1:L
            SSS = (q-1)*L^(-1) + rnd;
            while SSS > W1
                a = a + 1;
                W1 = W1 + Bfr_res_W(a);
            end
            Aft_res(b,:) = Bfr_res(a,:);
            % Aft_res: Pose after the resempling [1000x3]
            b = b + 1;
        end
        end
    for z = 1:Part_Num
        Particle.set(Aft_res(z,1), Aft_res(z,2),
Aft_res(z,3));
        Particle.plot(PAC,'particle');
    end

    % Estimation
    path_g(1,k+1) = mean(Aft_res(:,1)); % x path
    path_g(2,k+1) = mean(Aft_res(:,2)); % y path
    path_g(3,k+1) = mean(Aft_res(:,3)); % thata
end
```

```
    % plots
for k = 1:length(path_g)
    Robot5.set(path_e(1,k), path_e(2,k), path_e(3,k));
    Robot5_f.set(path_f(1,k), path_f(2,k), path_f(3,k));
    Robot5_g.set(path_g(1,k), path_g(2,k), path_g(3,k));
    Robot5_g.plot(RC, 'robot');
end
plot(path_g(1,:), path_g(2,:), 'Color','cyan','LineStyle',
'-.','LineWidth', 2);


% re-plot of sections 'e' and 'f' - due to the particles
for k = 1:5
    Robot5.plot(clrS(k,:), 'robot');
    Robot5_f.plot(clrS(k,:), 'robot');
    line([path_e(1,k) path_e(1,k+1)],[path_e(2,k)
path_e(2,k+1)],'Color',clrS(k,:),'LineStyle',
':','LineWidth', 2);
    line([path_f(1,k) path_f(1,k+1)],[path_f(2,k)
path_f(2,k+1)],'Color',clrS(k,:),'LineWidth', 2);
end
```

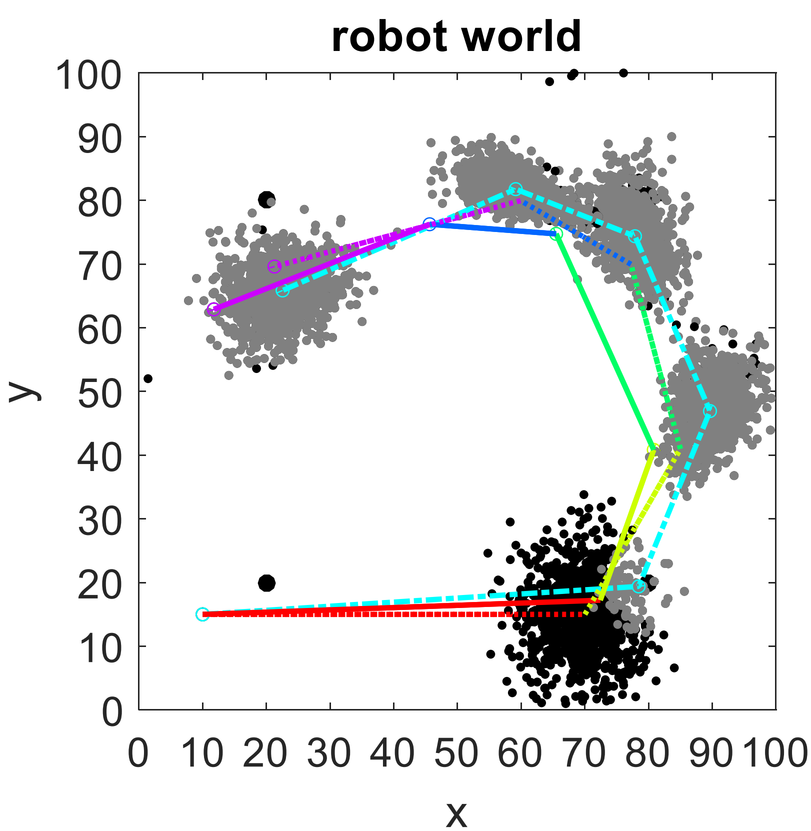

Figure 4 - The plot of true five robot poses and the robot path.

dotted line: *five robot poses and path resulting from the action command without noises* ***solid line:*** *true robot poses and path,* ***dash-dotted line:*** *implementation of particle filter. The robot poses and path colored in* ***cyan.***