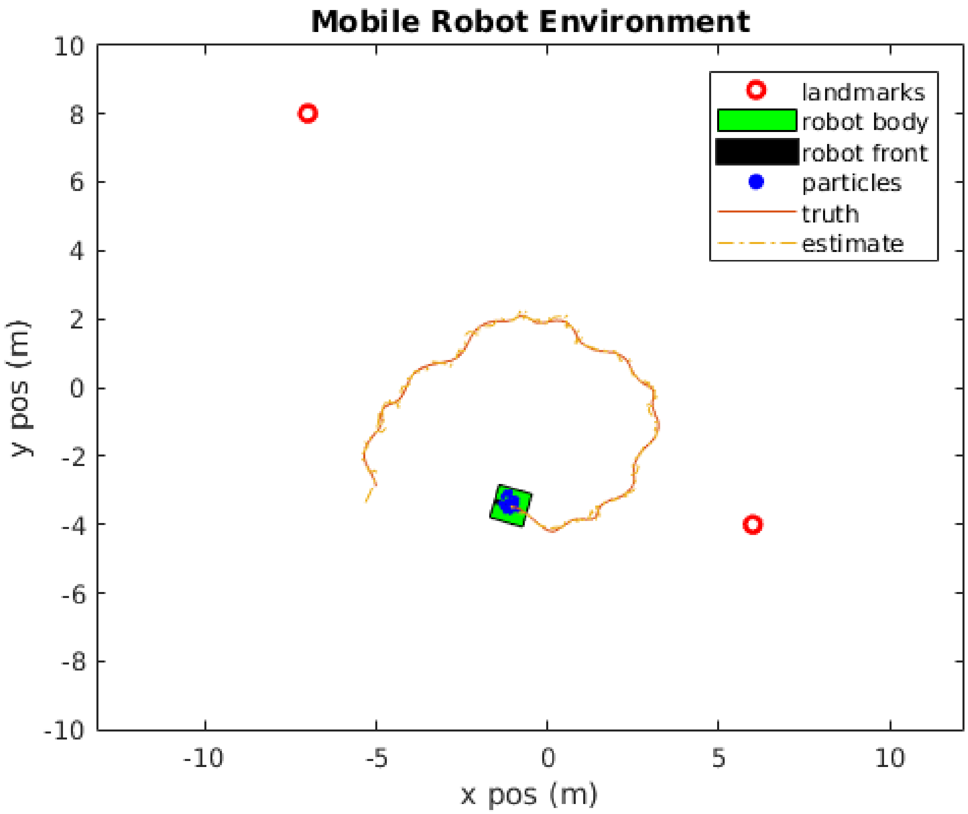
```
% Homework 4 Autonomous Two-wheel robot Particle Filter
% Jesse Wynn
% October 10, 2017

clc
clear
close all

% time params
Ts = 0.1;   % sec
t = 0:Ts:20;

% commanded velocity
v_c = 1 + 0.5 * cos(2 * pi * (0.2) * t);
w_c = -0.2 + 2 * cos(2 * pi * (0.6) * t);

% input noise
alpha_1 = 0.1;
alpha_4 = 0.1;
alpha_2 = 0.01;
alpha_3 = 0.01;
alpha_5 = 0.01;
alpha_6 = 0.01;

input_noise = [alpha_1, alpha_2, alpha_3, alpha_4, alpha_5, alpha_6];

% measurement noise
sigma_r = 0.1;
sigma_phi = 0.05;

% landmark locations
landmarks = [6, -7, 6;
             4, 8, -4];

num_landmarks = size(landmarks);
num_landmarks = num_landmarks(2);

% number of particles
M = 1000;

% initialize a bunch of random particles
Chi_t_prev = zeros(3, M);

% give them random x locations on the interval [-10 10]
Chi_t_prev(1,:) = -10 + (10 - (-10)).*rand(M,1);

% give them random y locations on the interval [-10 10]
Chi_t_prev(2,:) = -10 + (10 - (-10)).*rand(M,1);

% give them random headings on the interval [min_heading max_heading]
min_heading = pi/4;
max_heading = 3*pi/4;
```

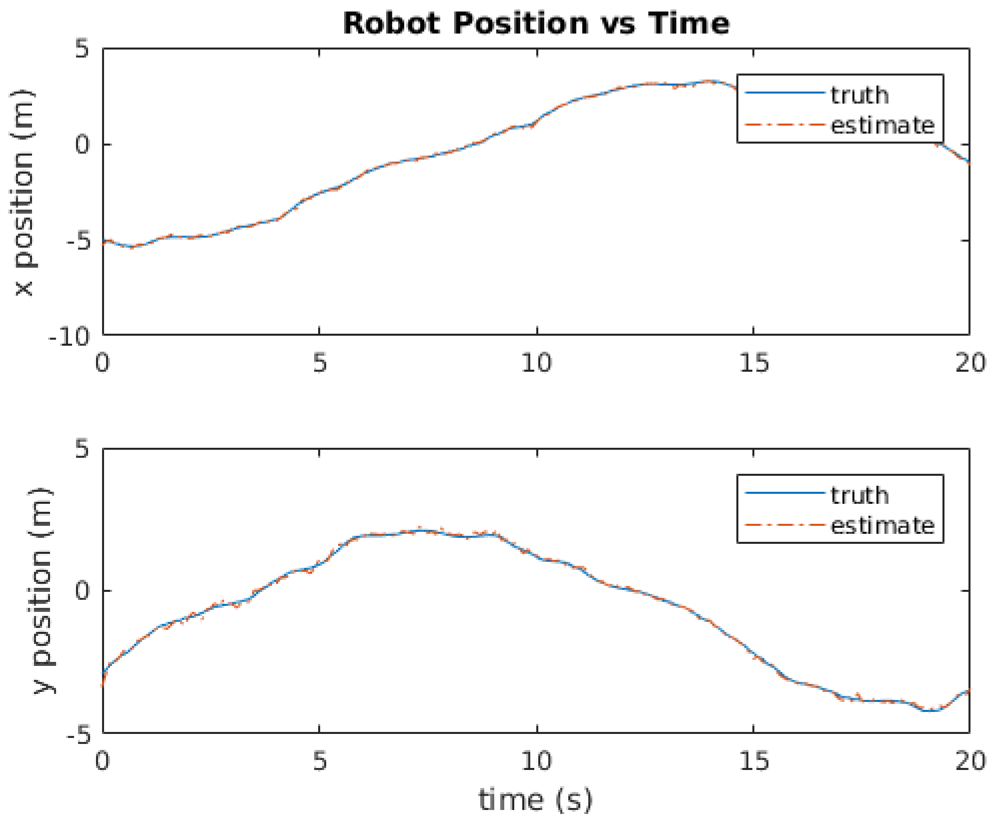
```
Chi_t_prev(3,:) = min_heading + (max_heading -
 min_heading).*rand(M,1);

% allocate space for holding data
x_true = zeros(1,length(t));
y_true = zeros(1,length(t));
theta_true = zeros(1,length(t));

Chi_bar_t = zeros(4,M);
W_t = zeros(1,M);
Chi_t = zeros(3,M);

range = zeros(length(t),num_landmarks);
bearing = zeros(length(t),num_landmarks);

x_est = zeros(1,length(t));
y_est = zeros(1,length(t));
theta_est = zeros(1,length(t));

Sigma_x = zeros(1,length(t));
Sigma_y = zeros(1,length(t));
Sigma_theta = zeros(1,length(t));

% robot inititial conditions
x = -5;
y = -3;
theta = pi/2;

% plot the first time
first = 0;
drawRobot(x,y,theta,landmarks, Chi_t_prev, first);
first = 1;
pause(0.5)

% loop through each time step
for i = 1:length(t)

    % implement velocity motion model from Table 5.3 (this gives us
 truth)
    v_hat = v_c(i) + randn*sqrt(alpha_1*(v_c(i))^2 +
 alpha_2*(w_c(i))^2);
    w_hat = w_c(i) + randn*sqrt(alpha_3*(v_c(i))^2 +
 alpha_4*(w_c(i))^2);
    gamma_hat = randn*sqrt(alpha_5*(v_c(i))^2 + alpha_6*(w_c(i))^2);
    x = x - (v_hat/w_hat)*sin(theta) + (v_hat/w_hat)*sin(theta +
 w_hat*Ts);
    y = y + (v_hat/w_hat)*cos(theta) - (v_hat/w_hat)*cos(theta +
 w_hat*Ts);
    theta = theta + w_hat*Ts + gamma_hat*Ts;

    % save off the true state data for plotting
    x_true(i) = x;
    y_true(i) = y;
    theta_true(i) = theta;
```

```
   % simulate range and bearing measurements to landmarks
   range(i,1) = norm([x; y] - landmarks(:,1)) + randn*sigma_r;   %
measurement + noise
   range(i,2) = norm([x; y] - landmarks(:,2)) + randn*sigma_r;
   range(i,3) = norm([x; y] - landmarks(:,3)) + randn*sigma_r;

   bearing(i,1) = atan2(landmarks(2,1) - y, landmarks(1,1) - x) -
theta + randn*sigma_phi;  % measurement + noise
   bearing(i,2) = atan2(landmarks(2,2) - y, landmarks(1,2) - x) -
theta + randn*sigma_phi;   % maybe sqrt
   bearing(i,3) = atan2(landmarks(2,3) - y, landmarks(1,3) - x) -
theta + randn*sigma_phi;

   % control input at time t
   u = [v_c(i); w_c(i)];

   % measurement for all three landmarks at time t
   z_t = [range(i,:);
          bearing(i,:)];

   % BEGIN particle filter (Monte Carlo Localization) algorithm
   % from table 8.2

   for m = 1:M
       % pass each particle through the motion model
       Chi_t(:,m) = sample_motion_model(u, Chi_t_prev(:,m),
input_noise, Ts);

       % assign a weight to each particle based on the current
measurement
       W_t(m) = measurement_model(z_t, Chi_t(:,m), landmarks,
[sigma_r, sigma_phi]);

       % augment the state of each particle with its associated
weight
       % Chi_bar_t(:,m) = [Chi_t(:,m); W_t(m)];  % instead of doing
this, just pass Chi_t and W_t into LVS
   end

   % use the low variance sampler to resample particles (particles
with
   % highter weight are more likely to be chosen and some will be
chosen
   % more than once.

   % first, normalize the weights
   W_t = W_t./sum(W_t);

   % then pass particles and weights to the LVS
   Chi_t_prev = LVS(Chi_t, W_t);

   % END particle filter algorithm
```

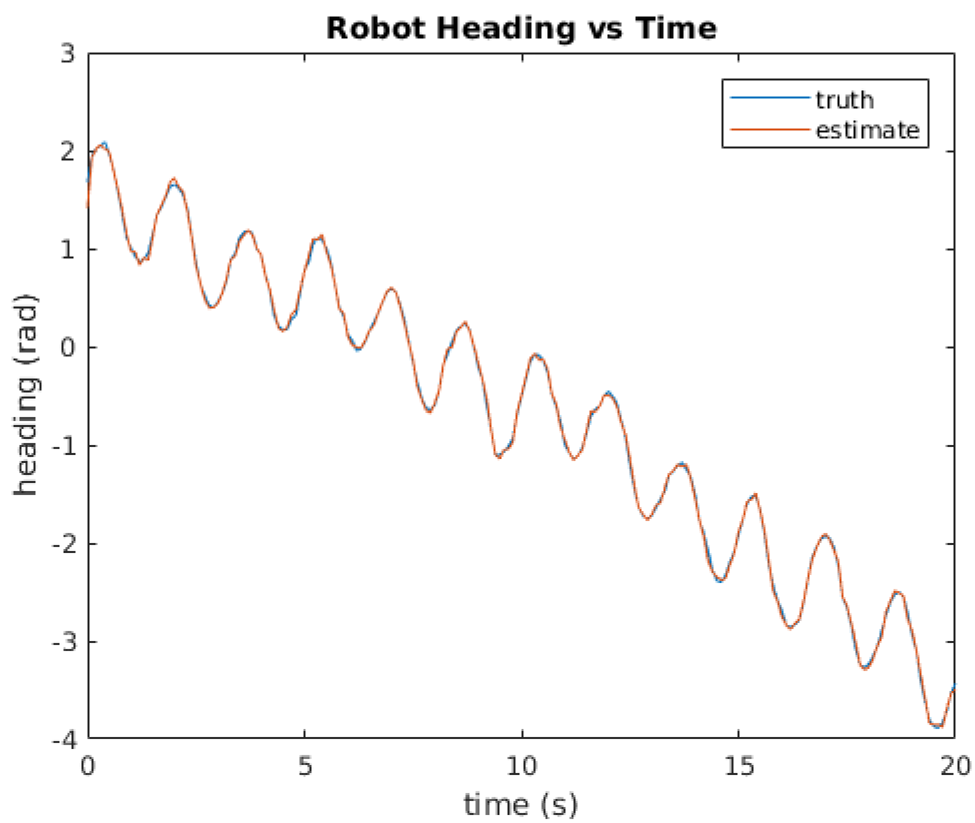
```
    % update the plot
    drawRobot(x,y,theta,landmarks, Chi_t_prev, first)
    pause(0.01)

    % estimates and error bounds
    x_est(i) = mean(Chi_t_prev(1,:));
    y_est(i) = mean(Chi_t_prev(2,:));
    theta_est(i) = mean(Chi_t_prev(3,:));

    Sigma_x(i) = std(Chi_t_prev(1,:));
    Sigma_y(i) = std(Chi_t_prev(2,:));
    Sigma_theta(i) = std(Chi_t_prev(3,:));
end

% plots
plot(x_true, y_true, x_est, y_est,'-.')
legend('landmarks','robot body','robot
 front', 'particles', 'truth','estimate')

figure(2), clf
subplot(2,1,1)
plot(t,x_true, t, x_est, '-.')
title('Robot Position vs Time')
ylabel('x position (m)')
legend('truth','estimate')

subplot(2,1,2)
plot(t,y_true, t, y_est, '-.')
ylabel('y position (m)')
xlabel('time (s)')
legend('truth','estimate')

figure(3), clf
plot(t,theta_true,t,theta_est)
title('Robot Heading vs Time')
xlabel('time (s)')
ylabel('heading (rad)')
legend('truth','estimate')

figure(4), clf
subplot(3,1,1)
plot(t,x_true - x_est,t,2*sqrt(Sigma_x),'r',t,-2*sqrt(Sigma_x),'r')
title('Error Plots')
ylabel('x')
legend('error','2-sigma bound')

subplot(3,1,2)
plot(t,y_true - y_est,t,2*sqrt(Sigma_y),'r',t,-2*sqrt(Sigma_y),'r')
ylabel('y')

subplot(3,1,3)
plot(t,theta_true -
 theta_est,t,2*sqrt(Sigma_theta),'r',t,-2*sqrt(Sigma_theta),'r')
ylabel('theta')
```

```
xlabel('time (s)')
```

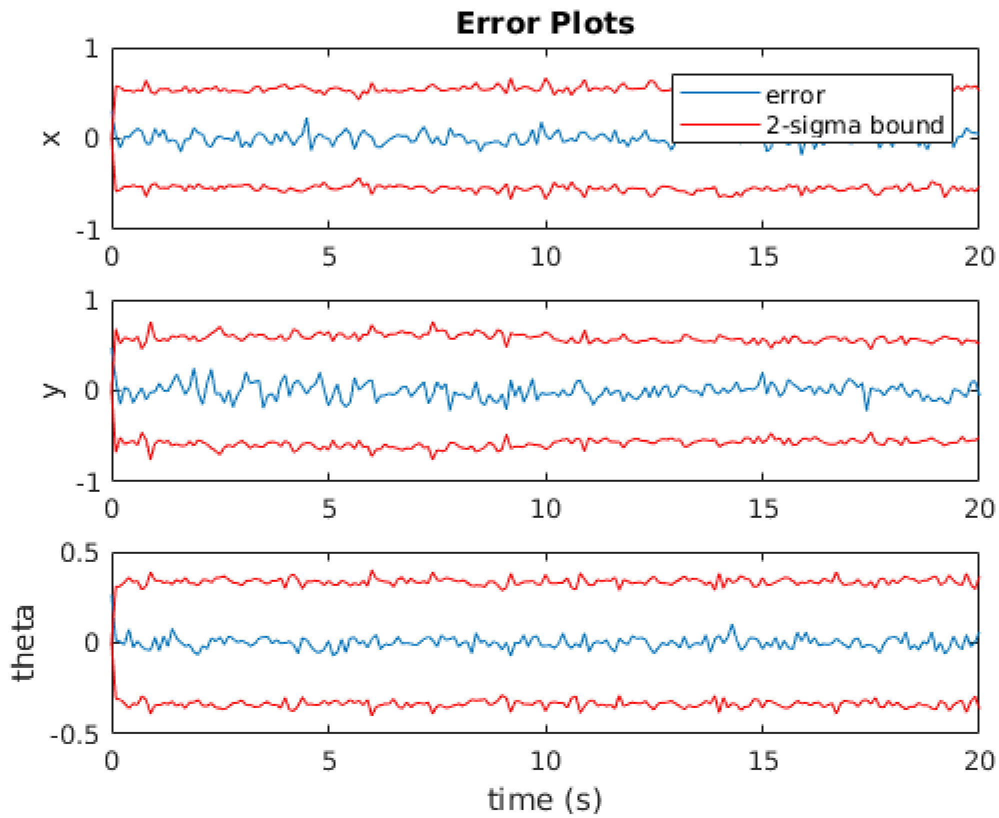




Robot Position vs Time
x position (m)
y position (m)
time (s)
truth
estimate



Robot Heading vs Time
heading (rad)
time (s)
truth
estimate



Published with MATLAB® R2017a