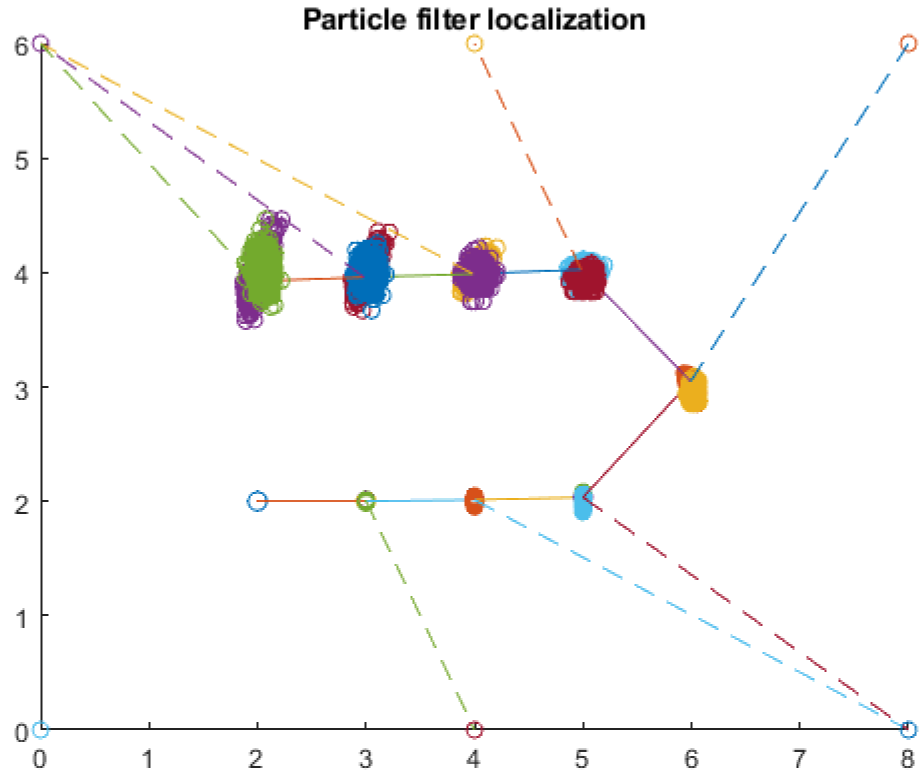
```
current_state=[2;2;0];
num_particles=1000;
[particles,weights]= initialization(current_state,num_particles);
% length(particles)
% length(weights)
velocity=[1 0;1 0;1 0;pi/2 pi/2;pi/2 pi/2;1 0;1 0; 1 0];
signature=[2;3;3;4;5;6;6;6];
alpha=[0.0001; 0.0001; 0.01; 0.0001; 0.0001; 0.0001];
scatter(current_state(1),current_state(2),50);
hold on
% scatter(particles(:,1),particles(:,2))
% hold on
%this data is for plot
statex=[2];
statey=[2];
for i = 1:8
    u(1)=velocity(i,1);
    u(2)=velocity(i,2);
    %disp(u)
    c=signature(i);
    z_mes=i;

 [next_state(1),next_state(2),next_state(3)]=motion_update(u,alpha,current_state);
   statex(end+1) = next_state(1);
   statey(end+1) =next_state(2);

 plot_mean(statex(length(statex)-1),statey(length(statey)-1),statex(length(statex)
    %next_state
    current_state=next_state;
    for  j= 1:length(particles)
      [upd_particles(j,1),upd_particles(j,2),upd_particles(j,3)]=
 motion_update(u,alpha,particles(j,:));
    end
    particles = upd_particles;
    scatter(current_state(1),current_state(2),50)
    %plot(current_state(1),current_state(2))
    hold on
    scatter(upd_particles(:,1),upd_particles(:,2))
    hold on
    weights=measurement_update(weights,z_mes,particles,c);
    y(:,1) =
 randsample(particles(:,1),length(particles),true,weights);
    y(:,2) =
 randsample(particles(:,2),length(particles),true,weights);
    y(:,3) =
 randsample(particles(:,3),length(particles),true,weights);

    scatter(y(:,1),y(:,2))

    hold on

end
```

```
plot_landmarks(statex,statey)

disp("Dear Grader, this is the implementation of particle filter, I
 have written functions for initialization , motion update and also
 measurement update. I have written some functions to plot. My code
 for plotting might not be the ideal one yet as i am new and learning
 matlab functions")
disp("I have plotted the mean of the motion to the robot, then the
 particles before update and also particles after the measurement
 update, the colors are a bit odd please ignore")


function line_plot = plot_mean(x1,y1,x2,y2)

plot([x1 x2],[y1 y2])
hold on
%line_plot=true

end


function plot_land = plot_landmarks(statex,statey)
title("Particle filter localization")
scatter(0,0)
hold on
scatter(4,0)
hold on
scatter(8,0)
hold on
scatter(8,6)
hold on
scatter(4,6)
hold on
scatter(0,6)
hold on
plot([statex(2) 4],[statey(2) 0],'--')
hold on
plot([statex(3) 8],[statey(3) 0],'--')
hold on
plot([statex(4) 8],[statey(4) 0],'--')
hold on
plot([statex(5) 8],[statey(5) 6],'--')
hold on
plot([statex(6) 4],[statey(6) 6],'--')
hold on
plot([statex(7) 0],[statey(7) 6],'--')
hold on
plot([statex(8) 0],[statey(8) 6],'--')
hold on
plot([statex(9) 0],[statey(9) 6],'--')
%plot_land=true

end
```

```
function [particles,weights] =
 initialization(current_state,num_particles)
    particles = zeros(num_particles,3);
    particles(:,1)=normrnd(current_state(1),0,[1,num_particles]);
    particles(:,2)=normrnd(current_state(2),0,[1,num_particles]);
    particles(:,3)=normrnd(current_state(3),0,[1,num_particles]);
    weights = zeros(num_particles,1)+(1/num_particles);
end

function [xprime,yprime,tethaprime]=motion_update(u,alpha,prev_state)
    v=u(1);
    w=u(2);
    x=prev_state(1);
    y=prev_state(2);
    tetha=prev_state(3);
    %disp("motionupdate")
    %disp(x)
    %disp(y)
    dt=1;
    vnoise = alpha(1)*v^2 + alpha(2)*w^2;
    wnoise = alpha(3)*v^2 + alpha(4)*w^2;
    gammanoise = alpha(5)*v^2 + alpha(6)*w^2;
    vhat = v+normrnd(0,vnoise);
    what = w+normrnd(0,wnoise);
    gammahat = 0+normrnd(0,gammanoise);
    k = vhat/what;
    xprime = x - k*sin(tetha)+k*sin(tetha+what*dt);
    yprime = y + k*cos(tetha)-k*cos(tetha+what*dt);
    tethaprime=tetha+what*dt+gammahat*dt;

end
%
function [weights] = measurement_update(weights,z,upd_particles,c)
eta =0;
mapObj = containers.Map({1,2,3,4,5,6},{[0.0;0.0],[4.0;0.0],[8.0;0.0],
[8.0;6.0],[4.0;6.0],[0.0;6.0]});
measurement=[2.276 5.249 2;4.321 5.834 3;3.418 5.869 3;3.774 5.911
 4;2.631 5.140 5;4.770 5.791 6;3.828 5.742 6;3.153 5.739 6];
sigmar=0.1;
sigmaphi=0.09;
landmark=mapObj(c);
mx=landmark(1);
my=landmark(2);
% z_measurement(1) = measurement(z,1);
% z_measurement(2) = measurement(z,2);
for i=1:length(upd_particles)
    x=upd_particles(i,1);
    y=upd_particles(i,2);
    tetha=upd_particles(i,3);
    rbar = sqrt(((mx-x)^2)+((my-y)^2));
    phibar=atan2(my-y,mx-x)-tetha;
    phibar=check_angle(phibar);
    range_error =measurement(z,1)-rbar;
    bearing_error = measurement(z,2)-phibar;
    prob_range = normpdf(range_error,0,sigmar);
    prob_bearing=normpdf(bearing_error,0,sigmaphi);
    upd = prob_range*prob_bearing;
    eta = eta + upd;
    weights(i)=upd;
end

weights=weights/eta;

end

function angle = check_angle(rad)
  if rad<0
      angle = rad + 2*pi;
  elseif rad >2*pi
       angle = rad- 2*pi;
  else
      angle = rad;
  end
end


% function xt = fun_sample_motion_model_velocity(ut,xt_1,alpha)
% given
% ut = control at time t
% xt_1 state at time t-1
% alpha noise 6 parameters
% dt = 1 time step duration
% n = no of samples to generate
% creating xt = 3 X n matrix of sampled states
% n=1000;
% xt = zeros(3,n);
% xnew=zeros(3,n);
% for i = 1:n
%     x = xt_1(1); %state variables
%     y = xt_1(2);
%     tetha=xt_1(3);
%     v = ut(1); %control variables
%     w = ut(2);
%     dt = 1;
%     adding noise
%
%     vnoise = alpha(1)*v^2 + alpha(2)*w^2;
%     wnoise = alpha(3)*v^2 + alpha(4)*w^2;
%     gammanoise = alpha(5)*v^2 + alpha(6)*w^2;
%     generating noise velocities
%     vhat = v+mvnrnd(0,vnoise);
%     what = w+mvnrnd(0,wnoise);
%     gammahat = 0+mvnrnd(0,gammanoise);
%     taking the ratio of linear to the angular velocity
%
%     k = vhat/what;
%     xprime = x - k*sin(tetha)+k*sin(tetha+what*dt);
%     yprime = y + k*cos(tetha)-k*cos(tetha+what*dt);
%     tethaprime=tetha+what*dt+gammahat*dt;
%     xt(:,i)=[xprime,yprime,tethaprime];
%
% end;
%
% end
%
%
% function xideal = motion_model_ideal(x0,u,dt)
%     x=x0(1);
%     y=x0(2);
%     tetha=x0(3);
%     v = u(1);
%     w = u(2);
%     r  = v/w;
%     xc = x+r*cos(tetha+pi/2);
%     yc = y+r*sin(tetha+pi/2);
%     xpr=xc+r*sin(tetha+w*dt);
%     ypr=yc-r*cos(tetha+w*dt);
%     tethapr =tetha+w*dt;
%     tetas = linspace(tetha,tethapr,10);
%     plot(xc+r*sin(tetas),yc-r*cos(tetas));
%     title('Scatter plot and trajectory for alpha2');
%     xideal=[xpr;ypr;tethapr];
%     draw(xideal,'g');
% end
```

Dear Grader, this is the implementation of particle filter, I have written functions for initialization , motion update and also measurement update. I have written some functions to plot. My code for plotting might not be the ideal one yet as i am new and learning matlab functions

I have plotted the mean of the motion to the robot, then the particles before update and also particles after the measurement update, the colors are a bit odd please ignore



Published with MATLAB® R2021a