```m
function [error] = particle_filter_localization(NP_out)
%PARTICLE_FILTER_LOCALIZATION Summary of this function goes here
%   Detailed explanation goes here

% ------------------------------------------------------------------------
% TASK for particle filter localization
% for robotic class in 2018 of ZJU

% Prepararion:
% 1. you need to know how to code and debug in matlab
% 2. understand the theory of Monte Carlo

% Then complete the code by YOURSELF!
% ------------------------------------------------------------------------

%close all;
%clear all;
NP_out=10;
disp('Particle Filter program start!!')

%% initialization
time = 0;
endTime = 60; % second
global dt;
dt = 0.1; % second

nSteps = ceil((endTime - time)/dt);

localizer.time = [];
localizer.xEst = [];
localizer.xGnd = [];
localizer.xOdom = [];
localizer.z = [];
localizer.PEst=[];
localizer.u=[];
localizer.error=[];
% Estimated State [x y yaw]'
xEst=[0 0 0]';
% GroundTruth State
xGnd = xEst;
% Odometry-only = Dead Reckoning
xOdom = xGnd;

% Covariance Matrix for predict
Q=diag([0.1 0.1 toRadian(3)]).^2;
% Covariance Matrix for observation
R=diag([1]).^2;% range:meter

% Simulation parameter
global Qsigma
Qsigma=diag([0.1 toRadian(5)]).^2;
global Rsigma
Rsigma=diag([0.1]).^2;

% landmark position
landMarks=[10 0; 10 10; 0 15; -5 20];


% longest observation confined
MAX_RANGE=20;
% Num of particles, initialized
NP=NP_out;
% Used in Resampling Step, a threshold
NTh=NP/2.0;

% particles produced
px=repmat(xEst,1,NP);
% weights of particles produced
pw=zeros(1,NP)+1/NP;


%% Main Loop

```

```
74
75  for i=1 : nSteps
76      %disp(size(px))
77      time = time + dt;
78      u=doControl(time);
79
80      % do observation
81      [z,xGnd,xOdom,u]=doObservation(xGnd, xOdom, u, landMarks, MAX_RANGE);
82      error=0;
83      for ip=1:NP
84
85          % process every particle
86          x=px(:,ip);
87          %w=1;
88          w=pw(:,ip);
89          dx=x(1)-xGnd(1);%误差计算
90          dy=x(2)-xGnd(2);
91          error=error+sqrt(dx^2+dy^2);
92          % do motion model and random sampling
93          x=doMotion(x, u)+sqrt(Q)*randn(3,1);
94           % calculate inportance weight
95          for iz=1:length(z(:,1))
96              pz=norm(x(1:2)'-z(iz,2:3));
97              dz=pz-z(iz,1);
98              w=w*Gaussian(dz,0,sqrt(R));
99          end
100         px(:,ip)=x;
101         pw(ip)=w;
102
103     end
104     error=error/NP;
105     pw=Normalization(pw,NP);
106     xEst=px*pw';
107     [px,pw]=ResamplingStep(px,pw,NTh,NP);
108
109
110
111     % Simulation Result
112     localizer.time=[localizer.time; time];
113     localizer.xGnd=[localizer.xGnd; xGnd'];
114     localizer.xOdom=[localizer.xOdom; xOdom'];
115     localizer.xEst=[localizer.xEst;xEst'];
116     localizer.u=[localizer.u; u'];
117      localizer.error=[localizer.error,error];
118     %Animation (remove some flames)
119     subplot(2,1,1)
120     if rem(i,10)==0
121         hold off;
122         arrow=0.5;
123         for ip=1:NP
124
                quiver(px(1,ip),px(2,ip),arrow*cos(px(3,ip)),arrow*sin(px(3,ip)),'ok');hol
                d on;
125         end
126         plot(localizer.xGnd(:,1),localizer.xGnd(:,2),'.b');hold on;
127         plot(landMarks(:,1),landMarks(:,2),'pk','MarkerSize',10);hold on;
128         if~isempty(z)
129             for iz=1:length(z(:,1))
130                 ray=[xGnd(1:2)';z(iz,2:3)];
131                 plot(ray(:,1),ray(:,2),'-r');hold on;
132             end
133         end
134         plot(localizer.xOdom(:,1),localizer.xOdom(:,2),'.k');hold on;
135         plot(localizer.xEst(:,1),localizer.xEst(:,2),'.r');hold on;
136         axis equal;
137         grid on;
138         drawnow;
139     end
140
141 end
142 error=localizer.error;
143 N=length(error);
144 error=sum(error)/N;
```

```
%
%draw the final results of localizer, compared to odometry & ground truth
drawResults(localizer);
subplot(2,1,2);
plot(localizer.error)
title('Average Error', 'fontsize', 12, 'fontname', 'times');
xlabel('Time(dt)', 'fontsize', 12, 'fontname', 'times');
ylabel('Average Error (m)', 'fontsize', 12, 'fontname', 'times');
end









%% Other functions

% degree to radian
function radian = toRadian(degree)
    radian = degree/180*pi;
end

function []=drawResults(localizer)
%Plot Result

    figure(1);
    hold off;
    x=[ localizer.xGnd(:,1:2) localizer.xEst(:,1:2)];
    set(gca, 'fontsize', 12, 'fontname', 'times');
    plot(x(:,1), x(:,2),'-.b','linewidth', 4); hold on;
    plot(x(:,3), x(:,4),'r','linewidth', 4); hold on;
    plot(localizer.xOdom(:,1), localizer.xOdom(:,2),'--k','linewidth', 4); hold on;

    title('Localization Result', 'fontsize', 12, 'fontname', 'times');
    xlabel('X (m)', 'fontsize', 12, 'fontname', 'times');
    ylabel('Y (m)', 'fontsize', 12, 'fontname', 'times');
    legend('Ground Truth','Particle Filter','Odometry Only');
    grid on;
    axis equal;

end

function [ u ] = doControl( time )
%DOCONTROL Summary of this function goes here
%   Detailed explanation goes here

    %Calc Input Parameter
    T=10; % [sec]

    % [V yawrate]
    V=1.0; % [m/s]
    yawrate = 5; % [deg/s]

    u =[ V*(1-exp(-time/T)) toRadian(yawrate)*(1-exp(-time/T))]';


end


%%  you need to complete

% do Observation model
function [z, xGnd, xOdom, u] = doObservation(xGnd, xOdom, u, landMarks, MAX_RANGE)
    global Qsigma;
    global Rsigma;

    % Gnd Truth and Odometry
    xGnd=doMotion(xGnd, u);% Ground Truth 理想状态
    u=u+sqrt(Qsigma)*randn(2,1);% add noise randomly
    xOdom=doMotion(xOdom, u); % odometry only
```

```
218
219         %Simulate Observation
220         z=[];
221         for iz=1:length(landMarks(:,1))
222             dx = xGnd(1)-landMarks(iz,1);
223             dy = xGnd(2)-landMarks(iz,2);
224             d=sqrt(dx^2+dy^2);
225             if d<MAX_RANGE
226                 z=[z;[d+sqrt(Rsigma)*randn(1,1) landMarks(iz,:)]];   % add observation
                    noise randomly
227             end
228         end
229     end
230
231
232     % do Motion Model
233     function [ x ] = doMotion( x, u)
234         global dt;
235
236         Delta = [ [dt*cos(x(3)+u(2)),0];
237                   [dt*sin(x(3)+u(2)),0];
238                   [0,dt]];
239
240         x = x+Delta*u;
241     end
242
243     % Gauss function
244     function g = Gaussian(x,u,sigma)
245         g=exp(-((x-u)^2)/((sigma^2)*2.0))/sqrt(2.0*pi*(sigma^2));
246     end
247
248     % Normalization
249     function pw=Normalization(pw,NP)
250         pw=pw/sum(pw);
251
252     end
253
254     % Resampling
255     function [px,pw]=ResamplingStep(px,pw,NTh,NP)
256         Neff=1.0/(pw*pw');
257         %Neff=0;
258         if Neff<NTh
259             ww=pw(1);
260             for iw=2:NP
261                 ww=[ww,ww(end)+pw(iw)];
262             end
263             pw1=[];
264             pp=[];
265             for i=1:NP
266                 r=rand();
267                 for j=1:NP
268                     if ww(j)>r
269                         pp=[pp,px(:,j)];
270                         pw1=[pw1,pw(:,j)];
271                         break
272                     end
273                 end
274             end
275             px=pp;
276             pw=pw1;
277         end
278     end
```