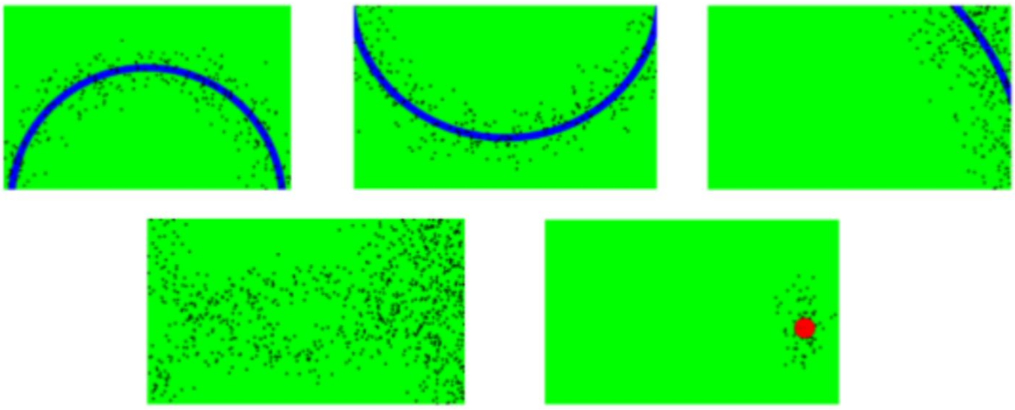
Homework8 Instruction

Problem 1: Given the following observation models, please use importance sampling and resampling techniques to estimate the robot location.

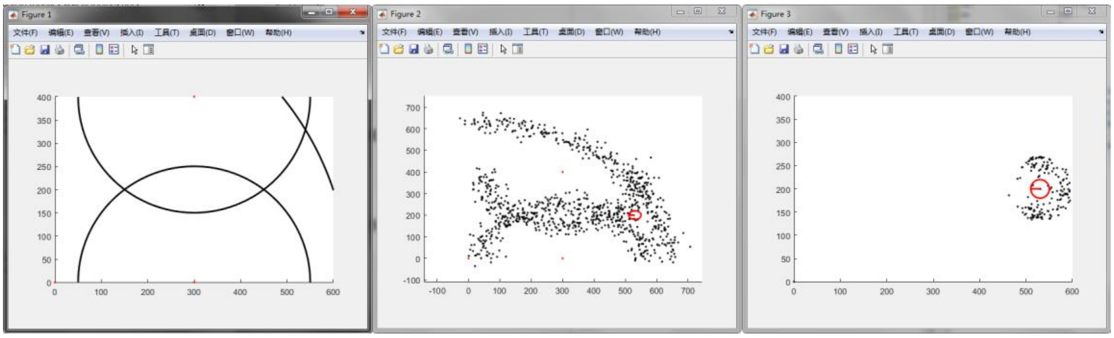


Algorithm Particle_filter($\mathcal{X}_{t-1}, u_t, z_t$):
$\quad \bar{\mathcal{X}}_t = \mathcal{X}_t = \emptyset$
$\quad$ *for* $m = 1$ *to* M *do*
$\quad\quad$ *sample* $x_t^{[m]} \sim p(x_t \mid u_t, x_{t-1}^{[m]})$
$\quad\quad$ $w_t^{[m]} = p(z_t \mid x_t^{[m]})$
$\quad\quad$ $\bar{\mathcal{X}}_t = \bar{\mathcal{X}}_t + \langle x_t^{[m]}, w_t^{[m]} \rangle$
$\quad$ *endfor*
$\quad$ *for* $m = 1$ *to* M *do*
$\quad\quad$ *draw* i *with probability* $\propto w_t^{[i]}$
$\quad\quad$ *add* $x_t^{[i]}$ *to* $\mathcal{X}_t$
$\quad$ *endfor*
$\quad$ *return* $\mathcal{X}_t$

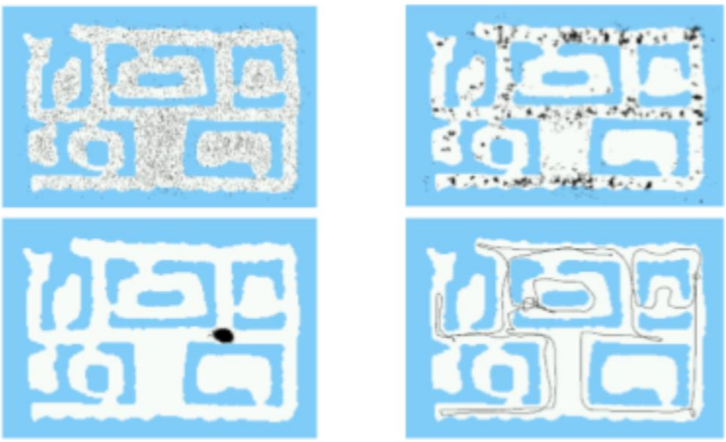
```
clc;clear;
close all;

% figure(1) is already generate
hold on
axis([0 600 0 400]);
set(gca,'PlotBoxAspectRatio',[6 4 1]);
```

```
r=250;
xc=300;
yc=0;
t = 0 : .01 : pi;
x = r * cos(t) + xc;
y = r * sin(t) + yc;
plot(x, y,'k','LineWidth',2)
plot(xc, yc,'.r','LineWidth',5);

xc=300;
yc=400;
t = 0 : .01 : pi;
x = r * cos(t) + xc;
y = -r * sin(t) + yc;
plot(x, y,'k','LineWidth',2)
plot(xc, yc,'.r','LineWidth',5);

r=632;
xc=0;
yc=0;
t = 0 : .01 : pi/2;
x = r * cos(t) + xc;
y = r * sin(t) + yc;
plot(x, y,'k','LineWidth',2)
plot(xc, yc,'.r','LineWidth',5);

figure(2) % figure(2) you should generate sample base on figure(1)
mu=
sigma=
hold on
axis([0 600 0 400]);
set(gca,'PlotBoxAspectRatio',[6 4 1]);

% generate you your sample point here

circleplot(530,200,20,pi) % drawing the robot

figure(3)
hold on
axis([0 600 0 400]);
set(gca,'PlotBoxAspectRatio',[6 4 1]);

% figure(3) implement your resampling algorithm
```

```
circleplot(m,n,20,pi)

function circleplot(xc, yc, r, theta)
t = 0 : .01 : 2*pi;
x = r * cos(t) + xc;
y = r * sin(t) + yc;
plot(x, y,'r','LineWidth',2)
t2 = 0 : .01 : r;
x = t2 * cos(theta) + xc;
y = t2 * sin(theta) + yc;
plot(x, y,'r','LineWidth',2)
end
```



Problem 2: Given a map and the ultrasound sensor model, please use importance sampling and resampling techniques to estimate the robot location and path.



```
rosinit
%% Load the Map of the Simulation World
filePath =
fullfile(fileparts(which('TurtleBotMonteCarloLocalizationExample')),'data','officemap.mat');
load(filePath);
show(map);
%% Setup the Laser Sensor Model and TurtleBot Motion Model
```

```
odometryModel = robotics.OdometryMotionModel;
odometryModel.Noise = [0.2 0.2 0.2 0.2];

rangeFinderModel = robotics.LikelihoodFieldSensorModel;
rangeFinderModel.SensorLimits = [0.45 8];
rangeFinderModel.Map = map;


% Query the Transformation Tree (tf tree) in ROS.
tftree = rostf;
waitForTransform(tftree,'/base_link','/camera_depth_frame');
sensorTransform = getTransform(tftree,'/base_link', '/camera_depth_frame');

% Get the euler rotation angles.
laserQuat = [sensorTransform.Transform.Rotation.W sensorTransform.Transform.Rotation.X ...
    sensorTransform.Transform.Rotation.Y sensorTransform.Transform.Rotation.Z];
laserRotation = quat2eul(laserQuat, 'ZYX');

% Setup the |SensorPose|, which includes the translation along base_link's
% +X, +Y direction in meters and rotation angle along base_link's +Z axis
% in radians.
rangeFinderModel.SensorPose = ...
    [sensorTransform.Transform.Translation.X sensorTransform.Transform.Translation.Y
laserRotation(1)];


%% Interface for Receiving Sensor Measurements From TurtleBot and Sending Velocity Commands to TurtleBot.
laserSub = rossubscriber('scan');
odomSub = rossubscriber('odom');

[velPub,velMsg] = ...
    rospublisher('/mobile_base/commands/velocity','geometry_msgs/Twist');

%% Initialize AMCL Object
amcl = robotics.MonteCarloLocalization;
amcl.UseLidarScan = true;

amcl.MotionModel = odometryModel;
amcl.SensorModel = rangeFinderModel;

amcl.UpdateThresholds = [0.2,0.2,0.2];
amcl.ResamplingInterval = 1;
```

```
%% Configure AMCL Object for Localization with Initial Pose Estimate.
amcl.ParticleLimits = [500 5000];
amcl.GlobalLocalization = false;
amcl.InitialPose = ExampleHelperAMCLGazeboTruePose();
amcl.InitialCovariance = eye(3)*0.5;

%% Setup Helper for Visualization and Driving TurtleBot.
visualizationHelper = ExampleHelperAMCLVisualization(map);

wanderHelper = ExampleHelperAMCLWanderer(laserSub, sensorTransform, velPub, velMsg);

%% Localization Procedure
numUpdates = 60;
i = 0;
while i < numUpdates
    % Receive laser scan and odometry message.
    scanMsg = receive(laserSub);
    odompose = odomSub.LatestMessage;

    % Create lidarScan object to pass to the AMCL object.
    scan = lidarScan(scanMsg);

    % For sensors that are mounted upside down, you need to reverse the
    % order of scan angle readings using 'flip' function.

    % Compute robot's pose [x,y,yaw] from odometry message.
    odomQuat = [odompose.Pose.Pose.Orientation.W, odompose.Pose.Pose.Orientation.X, ...
        odompose.Pose.Pose.Orientation.Y, odompose.Pose.Pose.Orientation.Z];
    odomRotation = quat2eul(odomQuat);
    pose = [odompose.Pose.Pose.Position.X, odompose.Pose.Pose.Position.Y odomRotation(1)];

    % Update estimated robot's pose and covariance using new odometry and
    % sensor readings.
    [isUpdated,estimatedPose, estimatedCovariance] = amcl(pose, scan);

    % Drive robot to next pose.
    wander(wanderHelper);

    % Plot the robot's estimated pose, particles and laser scans on the map.
    if isUpdated
        i = i + 1;
        plotStep(visualizationHelper, amcl, estimatedPose, scan, i)
    end
```

```
end

%% Stop the TurtleBot and Shutdown ROS in MATLAB
stop(wanderHelper);
rosshutdown
```