Autonomous Vehicle Simulation Using Reinforcement Learning (Avoid Obstcles)

BY

Tanshya Mishra(21BCI0233)

```
load exampleMaps.mat simpleMap
load exampleHelperOfficeAreaMap.mat office_area_map
mapMatrix = simpleMap;
mapScale = 1;
```

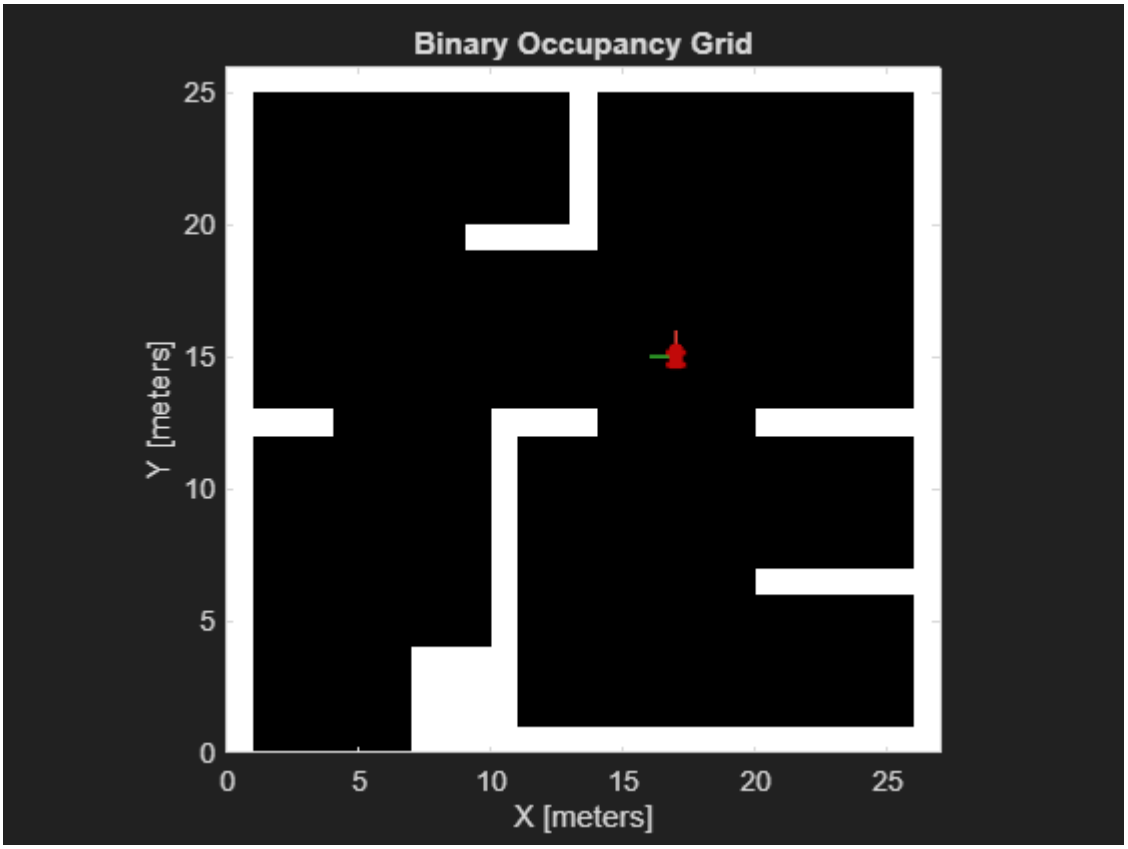
```
scanAngles = -3*pi/8:pi/8:3*pi/8;
maxRange = 12;
lidarNoiseVariance = 0.1^2;
lidarNoiseSeeds = randi(intmax,size(scanAngles));
```

```
% Max speed parameters
maxLinSpeed = 0.3;
maxAngSpeed = 0.3;

% Initial pose of the robot
initX = 17;
initY = 15;
initTheta = pi/2;
```

```
fig = figure("Name","simpleMap");
set(fig, "Visible","on");
ax = axes(fig);

show(binaryOccupancyMap(mapMatrix),"Parent",ax);
hold on
plotTransforms([initX,initY,0],eul2quat([initTheta,0,0]),"MeshFilePath","grou
ndvehicle.stl","View","2D");
light;
hold off
```




```
mdl = "HelperAvoidObstaclesMobileRobot";
Tfinal = 100;
sampleTime = 0.1;

agentBlk = mdl + "/Agent";
```

Open the model.

```
open_system(mdl)
```

```
Unable to find system or file 'HelperAvoidObstaclesMobileRobot'.
```

```
open_system(mdl + "/Environment")
```

```
obsInfo = rlNumericSpec([numel(scanAngles) 1],...
    "LowerLimit",zeros(numel(scanAngles),1),...
    "UpperLimit",ones(numel(scanAngles),1)*maxRange);
numObservations = obsInfo.Dimension(1);
```

```
numActions = 2;
actInfo = rlNumericSpec([numActions 1],...
    "LowerLimit",-1,...
    "UpperLimit",1);
```

```
env = rlSimulinkEnv(mdl,agentBlk,obsInfo,actInfo);
env.ResetFcn =
@(in)EHelperRLAvoidObstaclesResetFcn(in,scanAngles,maxRange,mapMatrix);
env.UseFastRestart = "Off";
```

```
statePath = [
    featureInputLayer(numObservations, "Normalization","none","Name","State")
    fullyConnectedLayer(50,"Name","CriticStateFC1")
    reluLayer("Name","CriticRelu1")
    fullyConnectedLayer(25,"Name","CriticStateFC2")];
actionPath = [
    featureInputLayer(numActions,"Normalization","none","Name","Action")
    fullyConnectedLayer(25,"Name","CriticActionFC1")];
commonPath = [
    additionLayer(2,"Name","add")
    reluLayer("Name","CriticCommonRelu")
    fullyConnectedLayer(1,"Name","CriticOutput")];

criticNetwork = layerGraph();
criticNetwork = addLayers(criticNetwork,statePath);
criticNetwork = addLayers(criticNetwork,actionPath);
criticNetwork = addLayers(criticNetwork,commonPath);
criticNetwork = connectLayers(criticNetwork,"CriticStateFC2","add/in1");
criticNetwork = connectLayers(criticNetwork,"CriticActionFC1","add/in2");
criticNetwork = dlnetwork(criticNetwork);
```

```
criticOptions =
rlOptimizerOptions("LearnRate",1e-3,"L2RegularizationFactor",1e-4,"GradientTh
reshold",1);
critic =
rlQValueFunction(criticNetwork,obsInfo,actInfo,"ObservationInputNames","State
","ActionInputNames","Action");
actorNetwork = [
    featureInputLayer(numObservations,"Normalization","none","Name","State")
    fullyConnectedLayer(50,"Name","actorFC1")
    reluLayer("Name","actorReLU1")
```

```
    fullyConnectedLayer(50, "Name","actorFC2")
    reluLayer("Name","actorReLU2")
    fullyConnectedLayer(2, "Name","actorFC3")
    tanhLayer("Name","Action")];
actorNetwork = dlnetwork(actorNetwork);

actorOptions =
rlOptimizerOptions("LearnRate",1e-4,"L2RegularizationFactor",1e-4,"GradientTh
reshold",1);
actor = rlContinuousDeterministicActor(actorNetwork,obsInfo,actInfo);
```

```
agentOpts = rlDDPGAgentOptions(...
    "SampleTime",sampleTime,...
    "ActorOptimizerOptions",actorOptions,...
    "CriticOptimizerOptions",criticOptions,...
    "DiscountFactor",0.995, ...
    "MiniBatchSize",128, ...
    "ExperienceBufferLength",1e6);

agentOpts.NoiseOptions.Variance = 0.1;
agentOpts.NoiseOptions.VarianceDecayRate = 1e-5;
```

```
obstacleAvoidanceAgent = rlDDPGAgent(actor,critic,agentOpts);
open_system(mdl + "/Agent")
```

```
maxEpisodes = 10000;
maxSteps = ceil(Tfinal/sampleTime);
trainOpts = rlTrainingOptions(...
    "MaxEpisodes",maxEpisodes, ...
    "MaxStepsPerEpisode",maxSteps, ...
    "ScoreAveragingWindowLength",50, ...
    "StopTrainingCriteria","AverageReward", ...
    "StopTrainingValue",400, ...
    "Verbose", true, ...
    "Plots","training-progress");
```

```
doTraining = false; % Toggle this to true for training.

if doTraining
```

```
    % Train the agent.
    trainingStats = train(obstacleAvoidanceAgent,env,trainOpts);
else
    % Load pretrained agent for the example.
    load HelperAvoidObstaclesAgent obstacleAvoidanceAgent
end
```

```
set_param("HelperAvoidObstaclesMobileRobot","StopTime","200");
out = sim("HelperAvoidObstaclesMobileRobot.slx");
```

```
for i = 1:5:size(out.range,3)
    u = out.pose(i,:);
    r = out.range(:,:,i);
    HelperAvoidObstaclesPosePlot(u,mapMatrix,mapScale,r,scanAngles,ax);
end
```

```
mapMatrix = office_area_map.occupancyMatrix > 0.5;
mapScale = 10;
initX = 20;
initY = 30;
initTheta = 0;
fig = figure("Name","office_area_map");
set(fig,"Visible","on");
ax = axes(fig);
show(binaryOccupancyMap(mapMatrix,mapScale),"Parent",ax);
hold on
plotTransforms([initX,initY,0],eul2quat([initTheta, 0,
0]),"MeshFilePath","groundvehicle.stl","View","2D");
light;
hold off
out = sim("HelperAvoidObstaclesMobileRobot.slx");
set_param("HelperAvoidObstaclesMobileRobot","StopTime","inf");
```

Visualize

```
for i = 1:5:size(out.range,3)
    u = out.pose(i,:);
    r = out.range(:,:,i);
    HelperAvoidObstaclesPosePlot(u,mapMatrix,mapScale,r,scanAngles,ax);
end
```