```
% clear all
% clc
```

Table of Contents

Locations of Obstacles 1
Initial position of 3 robot cleaners 1
Action space and Observation space........ 1
Create Robot Agents........ 2
    Policy Gradient Agent (with Baseline)........ 2
    Deep Q-Learning Network Agent........3
    Actor-Critic Agent........4
    Asynchronous Advantage Actor-Critic Agent........6
    Trust Region Policy Optimization........7
Training 8
Simulation........9

```
% Agentypes = { 'A3C','AC', 'DQN', 'PG','TRPO'};
% for j = 2:5
% RLAgentType = Agentypes{j};
```

Locations of Obstacles

```
obstaclesMat = [4 4; 5 4; 6 4; 7 4; 8 4; 9 4; 5 9; 6 9; 7 9; 8 11; 5 12; 6 12; 7 12; 8 12];
```

Initial position of 3 robot cleaners .

```
sAInit = [3 5];
sBInit = [10 5];
sCInit = [4 10];
s0 = [sAInit; sBInit; sCInit];
Tsample = 0.1; % sample time
Tfreq = 100; % simulation time
maxStep = ceil(Tfreq/Tsample); % Max Number of steps per episode
episode = 1; % initial training episodes
```

Simulink Model

```
model = "ConsolidatedModelV8";
```

Action space and Observation space

```
% Define observation space.
observationSize = [12 12 4];
obsInfo = rlNumericSpec(observationSize);
obsInfo.Name = 'ObservationsSpace';
```

```
% Define action space.
numAction = 8;
actionSpace = {1,2,3,4,5,6,7,8};
actInfo = rlFiniteSetSpec(actionSpace);
actInfo.Name = 'ActionsSapce';

blocks = model + ["/Robot Cleaner 1 (Red)","/Robot Cleaner 2 (Green)","/Robot Cleaner 3 (Blue)"
```

Create 2D grid world

```
env = rlSimulinkEnv(model,blocks,{obsInfo,obsInfo,obsInfo},{actInfo,actInfo,actInfo});
env.ResetFcn = @(in) resetMap(in, obstaclesMat);
```

Create Robot Agents

```
% random number generator
rng(0)
```

```
% episode =  [50 ;100]
% TypesAgent = ["PPO";"DQN";"AC" ]
% for  m  = 1:3
```

Choose Type of RL Agents

```
RLAgentType = inputdlg("Guide/ Agent 1 Reinforcement " + ...
    "Learning Type (TRPO/DQN/AC/PG/A3G): ", "Choices",[1 50])
```

```
RLAgentType =

  0×0 empty cell array
```

```
RLAgentType = cell2mat(RLAgentType);
  switch RLAgentType
```

Policy Gradient Agent (with Baseline)

```
      case "PG"
       baselineNetwork = [imageInputLayer(observationSize,'Normalization','none','Name','Observ
                            convolution2dLayer(16,32,'Name','Baseline2Dconv1' ,'WeightsInitializ
                            reluLayer('Name','RectifiedLU1')
                            convolution2dLayer(8,16,'Name','Baseline2Dconv2' ,'WeightsInitialize
                            reluLayer('Name','RectifiedLU2')
                            fullyConnectedLayer(512,'Name','BaselineFC1','WeightsInitializer','
                            reluLayer('Name','RectifiedLU3')
                            fullyConnectedLayer(256,'Name','BaselineFC2','WeightsInitializer','
                            reluLayer('Name','RectifiedLU4')
                            fullyConnectedLayer(128,'Name','BaselineFC3','WeightsInitializer','
                            reluLayer('Name','RectifiedLU5')
```

```
                             fullyConnectedLayer(1,'Name','BaselineFC4','WeightsInitializer','he
        baseOpt = rlOptimizerOptions('LearnRate',0.0001,'GradientThreshold',1,...
        'Algorithm','adam'...  % AMDAM Optimizer
          );
        actorNetwork = [imageInputLayer(observationSize,'Normalization','none','Name','Observati
                        convolution2dLayer(16,32,'Name','2Dconv1' ,'WeightsInitializer','he')
         reluLayer('Name','RectifiedLU1')
         convolution2dLayer(8,16,'Name','2Dconv2', 'WeightsInitializer','he')
         reluLayer('Name','RectifiedLU2')
         fullyConnectedLayer(512,'Name','FullyConnected1','WeightsInitializer','he')
         reluLayer('Name','RectifiedLU3')
         fullyConnectedLayer(256,'Name','FullyConnected2','WeightsInitializer','he')
         reluLayer('Name','RectifiedLU4')
         fullyConnectedLayer(128,'Name','FullyConnected3','WeightsInitializer','he')
         reluLayer('Name','RectifiedLU5')
         fullyConnectedLayer(numAction,'Name','ActorOutput')
         softmaxLayer('Name','action')]
        actorOpt = rlOptimizerOptions('LearnRate',0.0001,'GradientThreshold',1,...
            'Algorithm','adam'...  % AMDAM Optimizer
     );

        for i =1:3
           baseline(i) = rlValueFunction(baselineNetwork,obsInfo,'UseDevice',"gpu");
           actorNN(i) = rlDiscreteCategoricalActor(actorNetwork,obsInfo,actInfo,'UseDevice',"gpu
        end

        agentOpt =rlPGAgentOptions('UseBaseline',true, ...
                               'DiscountFactor',0.99, ...
                               'CriticOptimizerOptions',baseOpt, ...
                               'ActorOptimizerOptions',actorOpt);

        agent1 = rlPGAgent(actorNN(1),baseline(1),agentOpt);
        agent2 = rlPGAgent(actorNN(2),baseline(2),agentOpt);
        agent3 = rlPGAgent(actorNN(3),baseline(3),agentOpt);
```

Deep Q-Learning Network Agent

```
       case "DQN"

           for idx = 1:3
     % Create actorNN DNN.
     criticNetwork = [
         imageInputLayer(observationSize,'Normalization','none','Name','ObservationsInput')
         convolution2dLayer(16,32,'Name','2Dconv1' ,'WeightsInitializer','he')
         reluLayer('Name','RectifiedLU1')
         convolution2dLayer(8,16,'Name','2Dconv2', 'WeightsInitializer','he')
         reluLayer('Name','RectifiedLU2')
         fullyConnectedLayer(512,'Name','FullyConnected1','WeightsInitializer','he')
         reluLayer('Name','RectifiedLU3')
         fullyConnectedLayer(256,'Name','FullyConnected2','WeightsInitializer','he')
```

```
        reluLayer('Name','RectifiedLU4')
        fullyConnectedLayer(128,'Name','FullyConnected3','WeightsInitializer','he')
        reluLayer('Name','RectifiedLU5')
        fullyConnectedLayer(length(actInfo.Elements),'Name','CriticOutput')];

    criticDLNetwork = dlnetwork(criticNetwork);

    % create  criticNN
     criticNN(idx) = rlVectorQValueFunction(criticNetwork,obsInfo,actInfo,'UseDevice',"gpu");

    %criticNN optimizer options
          end
 DQNOptimOpts= rlOptimizerOptions('LearnRate',0.0001, ...
                                  'GradientThreshold',1, ...
                                  'Algorithm','adam'...  % AMDAM Optimizer
    );

agentOpt = rlDQNAgentOptions( ...
    'DiscountFactor',0.99,...
    'SampleTime',Tsample,... %event-based
    'UseDoubleDQN',false,...
    'CriticOptimizerOptions',DQNOptimOpts,... %ADAM Optimizer with learning rate 1e-4
    'ExperienceBufferLength',1e5,...  % replay buffer
    'MiniBatchSize',128, ... %sample batch size
    'TargetUpdateFrequency',10);

agentOpt.EpsilonGreedyExploration.EpsilonDecay = 0.0001;
% opt.EpsilonGreedyExploration.Epsilon = 0.9;
% opt.EpsilonGreedyExploration.EpsilonMin = 0.05;


% create DQN agents
agent1 = rlDQNAgent(criticNN(1),agentOpt);
agent2 = rlDQNAgent(criticNN(2),agentOpt);
agent3 = rlDQNAgent(criticNN(3),agentOpt);
```

Actor-Critic Agent

```
      case "AC"
for idx = 1:3
    % Create actor DNN.
    actorNetWork = [
        imageInputLayer(observationSize,'Normalization','none','Name','ObservationsInput')
        convolution2dLayer(16,32,'Name','2Dconv1' ,'WeightsInitializer','he')
        reluLayer('Name','RectifiedLU1')
        convolution2dLayer(8,16,'Name','2Dconv2', 'WeightsInitializer','he')
        reluLayer('Name','RectifiedLU2')
        fullyConnectedLayer(512,'Name','FullyConnected1','WeightsInitializer','he')
        reluLayer('Name','RectifiedLU3')
        fullyConnectedLayer(256,'Name','FullyConnected2','WeightsInitializer','he')
        reluLayer('Name','RectifiedLU4')
        fullyConnectedLayer(128,'Name','FullyConnected3','WeightsInitializer','he')
        reluLayer('Name','RectifiedLU5')
        fullyConnectedLayer(numAction,'Name','ActorOutput')
        softmaxLayer('Name','action')];
    actorDLNetWork = dlnetwork(actorNetWork);

    % Create critic DNN.
    criticNetwork = [
    imageInputLayer(observationSize,'Normalization','none','Name','ObservationsInput')
        convolution2dLayer(16,32,'Name','2Dconv1' ,'WeightsInitializer','he')
        reluLayer('Name','RectifiedLU1')
        convolution2dLayer(8,16,'Name','2Dconv2', 'WeightsInitializer','he')
        reluLayer('Name','RectifiedLU2')
        fullyConnectedLayer(512,'Name','FullyConnected1','WeightsInitializer','he')
        reluLayer('Name','RectifiedLU3')
        fullyConnectedLayer(256,'Name','FullyConnected2','WeightsInitializer','he')
        reluLayer('Name','RectifiedLU4')
        fullyConnectedLayer(128,'Name','FullyConnected3','WeightsInitializer','he')
        reluLayer('Name','RectifiedLU5')
        fullyConnectedLayer(1,'Name','CriticOutput')];
    criticDLNetwork = dlnetwork(criticNetwork);

    % create actorNN and criticNN
    actorNN(idx) = rlDiscreteCategoricalActor(actorDLNetWork,obsInfo,actInfo,'UseDevice',"gpu"
    criticNN(idx) = rlValueFunction(criticDLNetwork,obsInfo,'UseDevice',"gpu");
end

% actorNN and criticNN optimiser options .
actorOpts = rlOptimizerOptions('LearnRate',0.0001,'GradientThreshold',1, ...
    'Algorithm','adam'...  % AMDAM Optimizer
    );
criticOpts = rlOptimizerOptions('LearnRate',0.0001,'GradientThreshold',1, ...
    'Algorithm','adam'...  % AMDAM Optimizer
    );



agentOpt = rlACAgentOptions(...
    "NumStepsToLookAhead", 128, ...
   "EntropyLossWeight", 0.4, ...
    'ActorOptimizerOptions',actorOpts,...
    'CriticOptimizerOptions',criticOpts,...
    'SampleTime',Tsample,...
    'DiscountFactor',0.99);
% initOpt = rlAgentInitializationOptions;

% agents using the defined actors, critics, and options.
agent1 = rlACAgent(actorNN(1),criticNN(1),agentOpt);
```

```
agent2 = rlACAgent(actorNN(2),criticNN(2),agentOpt);
agent3 = rlACAgent(actorNN(3),criticNN(3),agentOpt);
```

Asynchronous Advantage Actor-Critic Agent

```
      case "A3C"

    % Create actorNN deep neural network.
    actorNetWork = [
        imageInputLayer(observationSize,'Normalization','none','Name','ObservationsInput')
        convolution2dLayer(16,32,'Name','2Dconv1' ,'WeightsInitializer','he')
        reluLayer('Name','RectifiedLU1')
        convolution2dLayer(8,16,'Name','2Dconv2', 'WeightsInitializer','he')
        reluLayer('Name','RectifiedLU2')
        fullyConnectedLayer(512,'Name','FullyConnected1','WeightsInitializer','he')
        reluLayer('Name','RectifiedLU3')
        fullyConnectedLayer(256,'Name','FullyConnected2','WeightsInitializer','he')
        reluLayer('Name','RectifiedLU4')
        fullyConnectedLayer(128,'Name','FullyConnected3','WeightsInitializer','he')
        reluLayer('Name','RectifiedLU5')
        fullyConnectedLayer(numAction,'Name','ActorOutput')
        softmaxLayer('Name','action')];
    actorDLNetWork = dlnetwork(actorNetWork);

    % Create criticNN deep neural network.
     criticNetwork = [
    imageInputLayer(observationSize,'Normalization','none','Name','ObservationsInput')
        convolution2dLayer(16,32,'Name','2Dconv1' ,'WeightsInitializer','he')
        reluLayer('Name','RectifiedLU1')
        convolution2dLayer(8,16,'Name','2Dconv2', 'WeightsInitializer','he')
        reluLayer('Name','RectifiedLU2')
        fullyConnectedLayer(512,'Name','FullyConnected1','WeightsInitializer','he')
        reluLayer('Name','RectifiedLU3')
        fullyConnectedLayer(256,'Name','FullyConnected2','WeightsInitializer','he')
        reluLayer('Name','RectifiedLU4')
        fullyConnectedLayer(128,'Name','FullyConnected3','WeightsInitializer','he')
        reluLayer('Name','RectifiedLU5')
        fullyConnectedLayer(1,'Name','CriticOutput')];
    criticDLNetwork = dlnetwork(criticNetwork);
for idx = 1:3
    % create actorNN and criticNN
    actorNN(idx) = rlDiscreteCategoricalActor(actorDLNetWork,obsInfo,actInfo,'UseDevice',"gpu"
    criticNN(idx) = rlValueFunction(criticDLNetwork,obsInfo,'UseDevice',"gpu");
end

% optimizer options for the actorNN and criticNN.
actorOpts = rlOptimizerOptions('LearnRate',0.0001,'GradientThreshold',1, ...
    'Algorithm','adam'...  % AMDAM Optimizer
    );
criticOpts = rlOptimizerOptions('LearnRate',0.0001,'GradientThreshold',1, ...
    'Algorithm','adam'...  % AMDAM Optimizer
    );

% A3C Agent Initialisation Options
initOpts = rlAgentInitializationOptions('UseRNN',true);

agentOpt = rlACAgentOptions("EntropyLossWeight", 0.4, ...
    'ActorOptimizerOptions',actorOpts,...
    'CriticOptimizerOptions',criticOpts,...
    'SampleTime',Tsample,...
    'DiscountFactor',0.99);

initOpt = rlAgentInitializationOptions('UseRNN',true);

% create AC agents
agent1 = rlACAgent(actorNN(1),criticNN(1),agentOpt,rlAgentInitializationOptions('UseRNN',true)
agent2 = rlACAgent(actorNN(2),criticNN(2),agentOpt,rlAgentInitializationOptions('UseRNN',true)
agent3 = rlACAgent(actorNN(3),criticNN(3),agentOpt,rlAgentInitializationOptions('UseRNN',true)

% agentA = rlACAgent(obsInfo(1),actInfo(1),rlAgentInitializationOptions('UseRNN',true));
% agentB = rlACAgent(obsInfo(2),actInfo(2),rlAgentInitializationOptions('UseRNN',true));
% agentC = rlACAgent(obsInfo(3),actInfo(3),rlAgentInitializationOptions('UseRNN',true));
```

Trust Region Policy Optimization

```
      case "TRPO"
for idx = 1:3
    % Create actorNN DNN.
      actorNetWork = [
        imageInputLayer(observationSize,'Normalization','none','Name','ObservationsInput')
        convolution2dLayer(16,32,'Name','2Dconv1' ,'WeightsInitializer','he')
        reluLayer('Name','RectifiedLU1')
        convolution2dLayer(8,16,'Name','2Dconv2', 'WeightsInitializer','he')
        reluLayer('Name','RectifiedLU2')
        fullyConnectedLayer(512,'Name','FullyConnected1','WeightsInitializer','he')
        reluLayer('Name','RectifiedLU3')
        fullyConnectedLayer(256,'Name','FullyConnected2','WeightsInitializer','he')
        reluLayer('Name','RectifiedLU4')
        fullyConnectedLayer(128,'Name','FullyConnected3','WeightsInitializer','he')
        reluLayer('Name','RectifiedLU5')
        fullyConnectedLayer(numAction,'Name','ActorOutput')
        softmaxLayer('Name','action')];
    actorDLNetWork = dlnetwork(actorNetWork);

    % Create criticNN DNN.
    criticNetwork = [
        imageInputLayer(observationSize,'Normalization','none','Name','ObservationsInput')
       convolution2dLayer(16,32,'Name','2Dconv1' ,'WeightsInitializer','he')
        reluLayer('Name','RectifiedLU1')
        convolution2dLayer(8,16,'Name','2Dconv2', 'WeightsInitializer','he')
        reluLayer('Name','RectifiedLU2')
        fullyConnectedLayer(512,'Name','FullyConnected1','WeightsInitializer','he')
        reluLayer('Name','RectifiedLU3')
        fullyConnectedLayer(256,'Name','FullyConnected2','WeightsInitializer','he')
        reluLayer('Name','RectifiedLU4')
        fullyConnectedLayer(128,'Name','FullyConnected3','WeightsInitializer','he')
        reluLayer('Name','RectifiedLU5')
        fullyConnectedLayer(1,'Name','CriticOutput')];
    criticDLNetwork = dlnetwork(criticNetwork);

    % create actorNN and criticNN
%     actorNN(idx) =  rlDiscreteCategoricalActor(actorNetWork,obsInfo,actInfo,'UseDevice',"gpu'
%     criticNN(idx) = rlValueFunction(criticNetwork,obsInfo,'UseDevice',"gpu");
  actorNN(idx)  = rlDiscreteCategoricalActor(actorDLNetWork,obsInfo,actInfo,'UseDevice',"gpu");
  criticNN(idx) = rlValueFunction(criticDLNetwork,obsInfo,'UseDevice',"gpu");
end

actorOpts = rlOptimizerOptions('LearnRate',0.0001,'GradientThreshold',1, ...
    'Algorithm','adam'...  % AMDAM Optimizer
    );
criticOpts = rlOptimizerOptions('LearnRate',0.0001,'GradientThreshold',1, ...
    'Algorithm','adam'...  % AMDAM Optimizer
    );

agentOpt = rlTRPOAgentOptions("AdvantageEstimateMethod","gae", ...
     'ExperienceHorizon',1024,...
    'CriticOptimizerOptions',criticOpts, ...
    'DiscountFactor',0.99, ...
    'ExperienceHorizon',maxStep,... % maxStep = 1000
    'SampleTime',Tsample,...
     'GAEFactor',0.95,...
     'EntropyLossWeight',0.01,...
    'MiniBatchSize',64,...
    'NumEpoch',3);

agent1 = rlTRPOAgent(actorNN(1),criticNN(1),agentOpt);
agent2 = rlTRPOAgent(actorNN(2),criticNN(2),agentOpt);
agent3 = rlTRPOAgent(actorNN(3),criticNN(3),agentOpt);
```

```
% End of Switch
  end
```

```
SWITCH expression must be a scalar or a character vector.
```

Training

Training Options

```
if RLAgentType == "DQN"
    trainOpts = rlMultiAgentTrainingOptions(...
    "AgentGroups",{[1,2,3]},...
    "LearningStrategy","centralized",...
    'MaxEpisodes',50,...
    'maxStepPerEpisode',maxStep,...
    'Plots','training-progress',...
    'ScoreAveragingWindowLength',120,...
    'StopTrainingCriteria','EpisodeCount',...
    'StopTrainingValue',100);
else
    % TRPO, AC,A3C,PG training is compatible with CTDE strategy
    trainOpts = rlMultiAgentTrainingOptions(...
    "LearningStrategy","decentralized",...
    'MaxEpisodes',50,...
    'maxStepPerEpisode',maxStep,...
    'Plots','training-progress',...
    'ScoreAveragingWindowLength',120,...
    'StopTrainingCriteria','EpisodeCount',...
    'StopTrainingValue',100);
end
```

Start Training

```
startTraining = inputdlg("Do training (true or false): ", "Choices",[1 50])
startTraining = cell2mat(startTraining);
% doTraining = true;
if startTraining
results = train([agent1,agent2,agent3],env,trainOpts);
```

Simulation

```
else
loadStr = ['50',RLAgentType, 'AgenttrainedAgents.mat'];
load(loadStr);
rng(0) % reset the random generator
simulateOpts = rlSimulationOptions('maxStep',maxStep);
experiences = sim(env,[agent1,agent2,agent3],simulateOpts);
end
```

save train or simulaiton results

```
if startTraining
    str = [num2str(trainOpts.MaxEpisodes),RLAgentType, 'trainedAgents.mat'];
    save(str,'result')
else
    str2 = [RLAgentType,'simExp.mat'];
    save(str2,'experience')
end
```

```
% %end for
% end
```