强化学习下阿克曼小车自动泊车的 MATLAB 仿真

本项目旨在以 MATLAB 软件模拟一辆阿克曼结构的小车运动学参数，实现强化学习下的垂直倒车入库。在之前的两个项目中，本文已经对垂直倒车轨迹及垂直倒车过程具有一定的了解，下面本文将基于前文倒车入库的轨迹和流程，分步实现 PPO 算法下的 MATLAB 仿真。并对强化学习的训练及测试情况进行展示。

一、 PPO 算法

在介绍本文的强化学习 MATLAB 项目前，先介绍一下 MathWorks 中的强化学习停车技术：Train PPO Agent for Automatic Parking Valet。其主要面对的是策略不稳定、数据效率低的问题。因为，在许多策略梯度方法中，由于步长较大，策略更新不稳定，导致错误的策略更新，当这个新的错误策略被用于学习时，会导致更糟糕的策略。如果步骤很小，那么就会导致学习的缓慢。而且很多学习方法都是借鉴现有经验，在梯度更新后丢弃经验。这使得学习过程变慢，因为神经网络需要大量的数据来学习。

PPO 即 Proximal Policy Optimization 意为近端策略优化算法，是强化学习中最经典的博弈对抗算法，在当下的梯度法中，本文不用现行 Policy 的日志，而是用现行 Policy 与旧 Policy 的比率：

$$\hat{\mathbb{E}}_t\left[\frac{\pi_\theta(a_t \mid s_t)}{\pi_{\theta_{\text{old}}}(a_t \mid s_t)}\hat{A}_t\right] = \hat{\mathbb{E}}_t\left[r_t(\theta)\hat{A}_t\right]$$

$$L^{CLIP}(\theta) = \hat{\mathbb{E}}_t\left[\min(r_t(\theta)\hat{A}_t, \text{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t)\right]$$

最后本文的目标包含三个部分，分别是 Lclip、MSE 即预测的状态值和目标的平方损失以及用来鼓励探索的熵。其算法实现步骤大概分为五步进行：

STEP 1：游戏 n 步，存储状态，动作概率，奖励，完成变量。

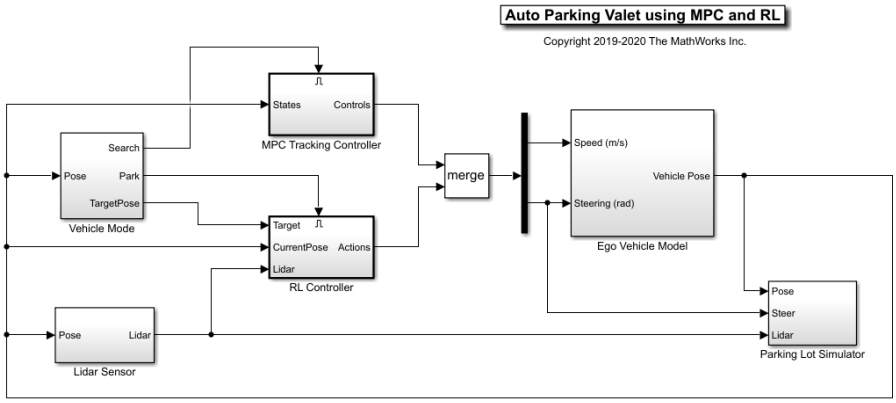
STEP 2：基于上述经验，应用广义优势估计方法。本文将在编码部分看到这一点。

STEP 3：通过计算各自的损失，训练神经网络在某些时期的运行。

STEP 4：对完成训练的模型测试 m 轮。

STEP 5：如果测试片段的平均奖励大于你设定的目标奖励，随即停止，否则就从第一步开始重复。

在 MATLAB 中其 Simulink 模型如下：



该模型中的自主车辆动力学由具有两个输入的单轨自行车模型表示：车速（m/s）和转向角（弧度），输出车辆位置和姿态。MPC 和 RL 控制器放置在“启用子系统”模块中，该模块由表示车辆是否必须搜索空位或执行停车操作的信号激活。启用信号由“车辆模式”子系统中的“摄像头”算法确定。最初，车辆处于搜索模式，MPC 控制器跟踪参考路径。当找到空位时，将激活停车模式，并且 RL 代理执行停车动作。

二、MathWorks 中的 PPO

在 MATLAB 已有的框架中，Train PPO Agent for Automatic Parking Valet 含有多个功能完善，效果良好的功能函数：

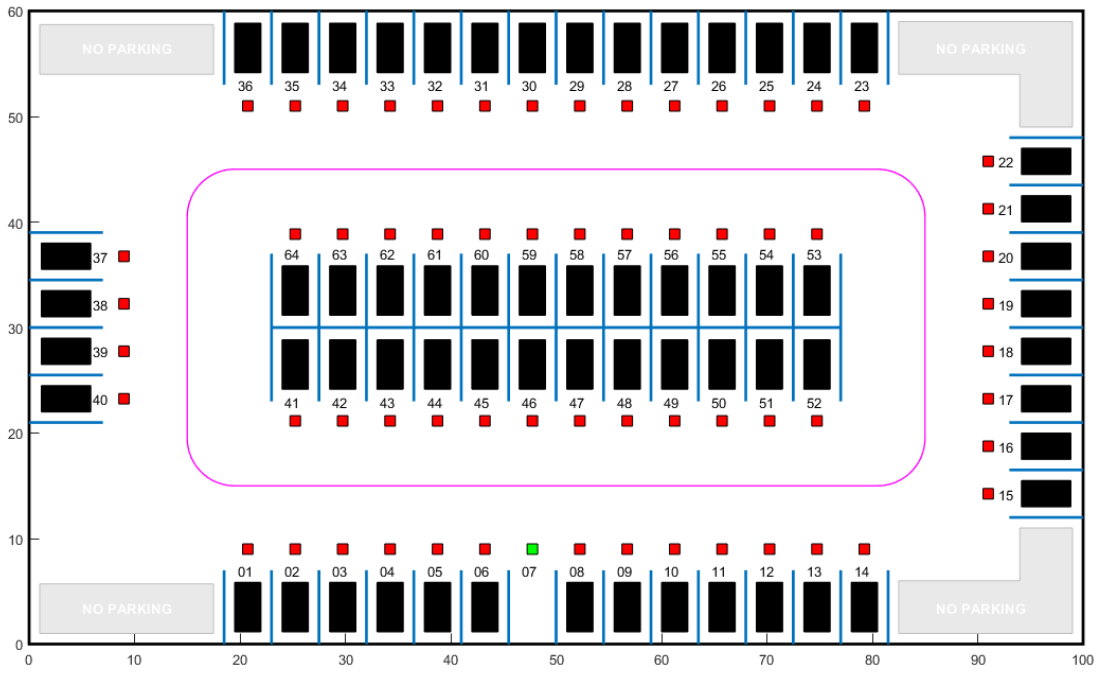
- autoParkingValetParams.m
- autoParkingValetResetFcn.m
- Camera.m
- createMPCForParking.m
- getCarSegmentLengths.m
- getRefTraj.m
- lidarSegmentIntersections.m
- LIDARSensor.m
- ParkingLotSimulator.m
- parkingVehicleStateFcnRRT.m
- parkingVehicleStateJacobianFcnRRT.m
- rect2segs.m
- vehicleStateFcn.m
- vehicleStateJacobianFcnDT.m

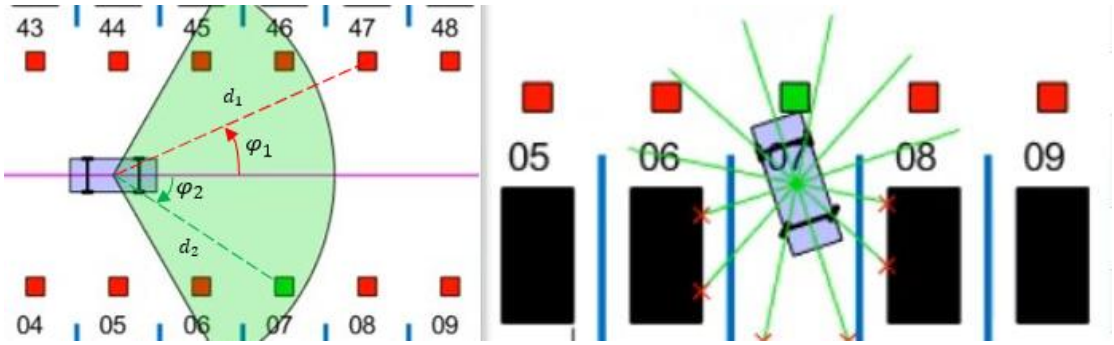
MathWorks 下的 Train PPO Agent for Automatic Parking Valet 官方网址为：
https://www.mathworks.com/help/releases/R2020b/reinforcement-learning/ug/train-ppo-agent-for-automatic-parking-valet.html

以 LIDARSensor.m 问件为例，其内是激光雷达传感器的功能函数，下面是对代码内容的展示：

```
classdef LIDARSensor < matlab.System & ...
        matlab.system.mixin.Propagates
% Lidar sensor model

% Copyright 2020 The MathWorks, Inc.

    properties
        % Rectangles global position and orientation specified as a N x 5
        % vector [x,y,L,W,theta]
        %Rectangles = [5,5,1,3,0;5,-5,1,3,0]
    end
    properties (Nontunable)
        % MapObject Name of ParkingLot object
        MapObject = ''

        % Geometry of the car [L,W]
        CarGeometry = [3,1]

        % Max distance reading of the sensor
        MaxDistance = 12
    end
    properties (Nontunable,PositiveInteger)
        % Number of sensor readings available
        SensorResolution = 60; %numObsLidar
    end
```

函数所在文件中都有对代码的详细解释与介绍，上图所示即为传感器的矩形全局位置和方向设置。其项目整体效果如下：





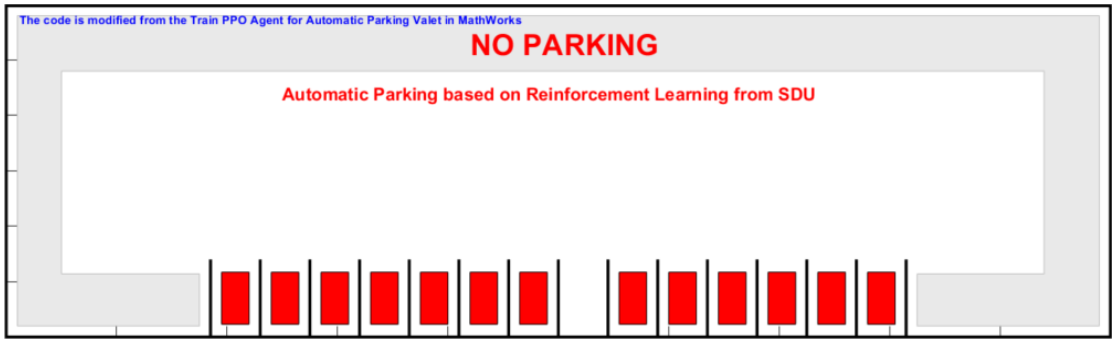
（PS：具体其实现方式和操作指令请进入官方网站即可得到相关信息）

三、基于 PPO 的垂直倒车入库 RL 实现

与本次项目要求不同的是，该项目的停车，是西方所留行的车头入库方式，是故根据项目所要求的垂直倒车入库，本文对其行驶方式进行了修改，并进行搭建停车场、传感器和摄像头等的创新和改变，其具体实施方式是对 main 函数进行编写以及对 ParkingLot 函数进行较大的修改。

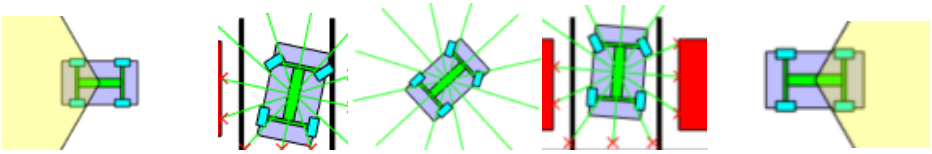
在通过先前的泊车轨迹代码复刻和垂直倒车入库原理的复现之后，本文进行基于近端策略优化算法下的强化学习自动泊车软件模拟的实现。

首先，本文基于项目需要在 ParkingLot 函数中对停车场构型进行重新架构，本文取消了原函数中的对称性停车场，而是适应性的对整个环境进行调整，这个结果如下所示：

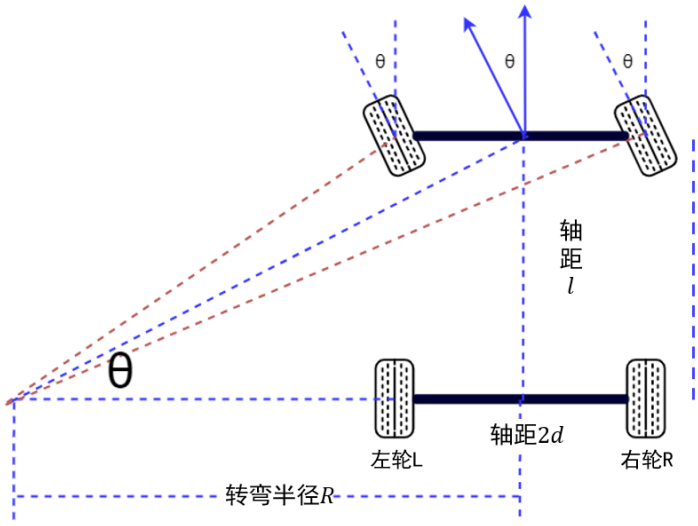


在倒车场景中本文设计了并排车位十四处，并在车位外的边界环境都设置有 No Parking 的区域，其目的是约束小车在训练后不会停在错误的地方。此外，红色字体标注为本次 MATLAB 模拟项目：Automatic Parking based on Reinforcement Learning from SDU。蓝色字体则是注明，本项目框架是由 MathWorks 中的开源文件 Train PPO Agent for Automatic Parking Valet 而来。

而在小车结构中，本文基于阿克曼小车结构，对小车模型进行了修改，以满足阿克曼小车的结果要求，



可以看出图中的所还原的是阿克曼转向结构，其底盘类似现实中真实的汽车底盘。底盘前两轮用于控制小车运动方向，后两轮用于控制其速度。

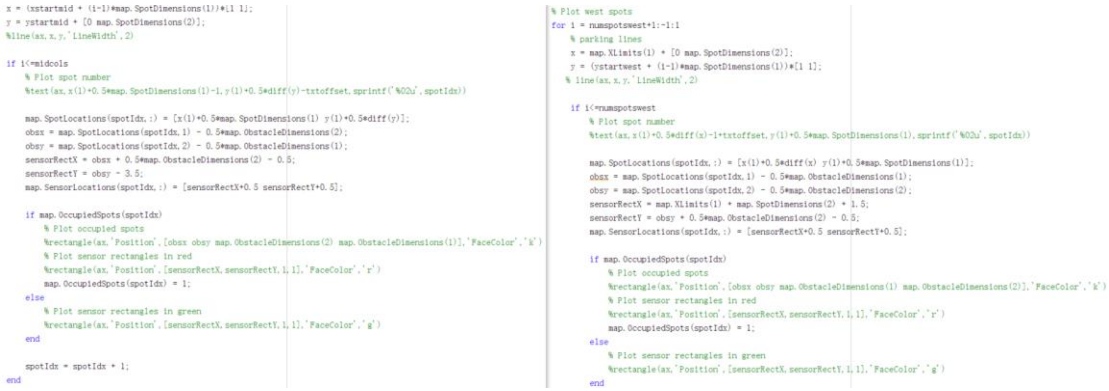


此外，本文对传感器范围，摄像头范围等具体实施参数进行了修改，以贴合自动泊车项目的实施需要。

值得一提的是，本文在进行自主修改的 ParkingLot 函数的代码文件中，修改部分都进行了相应注释，即便读者对 MATLAB 和强化学习了解程度一般也能较快速理解。

```
% Plot north spots
for i = numspotsnorth+1:-1:1
    % parking lines
    x = (xstartnorth + (i-1)*map.SpotDimensions(1))*[1 1];
    y = map.YLimits(2) + [-map.SpotDimensions(2) 0];
    %line(ax,x,y,'LineWidth',2)

    if i<=numspotsnorth
        % Plot spot number
        %text(ax,x(1)+0.5*map.SpotDimensions(1)-1,y(1)+0.5*diff(y)-txtoffset,sprintf('%02d',spotIdx))

        map.SpotLocations(spotIdx,:) = [x(1)+0.5*map.SpotDimensions(1) y(1)+0.5*diff(y)];
        obsx = map.SpotLocations(spotIdx,1) - 0.5*map.ObstacleDimensions(2);
        obsy = map.SpotLocations(spotIdx,2) - 0.5*map.ObstacleDimensions(1);
        sensorRectX = obsx + 0.5*map.ObstacleDimensions(2) - 0.5;
        sensorRectY = map.YLimits(2) - map.SpotDimensions(2) - 2.5;
        map.SensorLocations(spotIdx,:) = [sensorRectX+0.5 sensorRectY+0.5];

        if map.OccupiedSpots(spotIdx)
            % Plot occupied spots
            %rectangle(ax,'Position',[obsx obsy map.ObstacleDimensions(2) map.ObstacleDimensions(1)],'FaceColor','k')
            % Plot sensor rectangles in red
            %rectangle(ax,'Position',[sensorRectX sensorRectY 1 1],'FaceColor','r')
            map.OccupiedSpots(spotIdx) = 1;
        else
            % Plot sensor rectangles in green
            %rectangle(ax,'Position',[sensorRectX sensorRectY 1 1],'FaceColor','g')
        end
```

```
x = (xstartsouth + (i-1)*map.SpotDimensions(1))*[1 1];
y = map.YLimits(1) + [0 map.SpotDimensions(2)];
line(ax,x,y,'LineWidth',2,'color','k')

if i<=numspotssouth  % to avoid plotting an extra vehicle at the end
    % Plot spot number
    %text(ax,x(1)+0.5*map.SpotDimensions(1)-1,y(1)+0.5*diff(y)+txtoffset,sprintf('%02d',spotIdx))

    map.SpotLocations(spotIdx,:) = [x(1)+0.5*map.SpotDimensions(1) y(1)+0.5*diff(y)];
    obsx = map.SpotLocations(spotIdx,1) - 0.5*map.ObstacleDimensions(2);
    obsy = map.SpotLocations(spotIdx,2) - 0.5*map.ObstacleDimensions(1);
      obsxx = map.SpotLocations(spotIdx,1) + 0.1*map.ObstacleDimensions(2);
      obsyy = map.SpotLocations(spotIdx,2) + 0.1*map.ObstacleDimensions(1);
    sensorRectX = obsx + 0.5*map.ObstacleDimensions(2) - 0.5;
    sensorRectY = map.YLimits(1) + map.SpotDimensions(2) + 2 - 0.5;
    map.SensorLocations(spotIdx,:) = [sensorRectX+0.5 sensorRectY+0.5];

    if map.OccupiedSpots(spotIdx)

        % Plot occupied spots
        rectangle(ax,'Position',[obsx obsy map.ObstacleDimensions(2) map.ObstacleDimensions(1)],'FaceColor','r') % color of other cars
        % Plot sensor rectangles in red   %传感器矩形
        %rectangle(ax,'Position',[sensorRectX sensorRectY 1 1],'FaceColor','r')
        map.OccupiedSpots(spotIdx) = 1;
    else
        % Plot sensor rectangles in green
        %rectangle(ax,'Position',[sensorRectX sensorRectY 1 1],'FaceColor','g')
    end
```



（部分注释如上所示）

与原项目过程中，本文对小车的仿真设备位置等功能进行了改变，最大的变化是将小车由车头倒车改为车尾倒车入库。其次是对传感器感应距离进行了调整，考虑到本文的停车场环境较 MathWorks 中的环境较为简单清晰，本文通过缩短感应距离以减少繁琐的信息处理。

在 PPO 的架构中，最多设置 10000 个情节，每个情节最多持续 200 个时间步长。

当达到最大情节数目或超过 100 个情节的平均奖励超过 100 时，训练终止。使用 rlTrainingOptions 对象指定训练选项。

在训练过程中，针对算法实现过程中的具体情况，本文对部分参数进行优化以改善小车泊车时车身的抖动现象，并且在训练次数为 1000 次这样体量较小的训练情况下也有较好训练结果。

```
%Specify the options for training using an object
trainOpts = rlTrainingOptions(...
    'MaxEpisodes',1000,...
    'MaxStepsPerEpisode',200,...
    'ScoreAveragingWindowLength',200,...
    'Plots','training-progress',...
    'StopTrainingCriteria','AverageReward',...
    'StopTrainingValue',80);
```

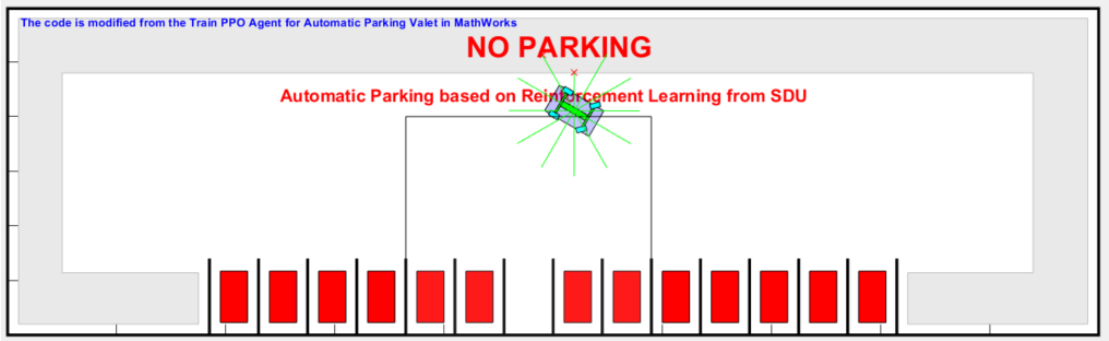
在 main 函数中，本文先指定小车所在位置及相应位姿，并设置训练场景下的空余车位，再使用脚本创建自适应 MPC 控制器对象以进行参考轨迹跟踪。进一步本文创建 Simulink 环境接口，指定 RL 代理块的路径。并指定用于训练的重置函数。该函数在每集开始时将自我载体的初始姿势重置为随机值。

```
freeSpotIdx = 8;
map = ParkingLot(freeSpotIdx);
egoInitialPose = [80, 15, pi];
egoTargetPose = createTargetPose(map,freeSpotIdx);
autoParkingValetParams
mdl = 'rlAutoParkingValet';
%open_system(mdl)
createMPCForParking
%Create observation and operation specifications for the environment
numObservations = 16;
observationInfo = rlNumericSpec([numObservations 1]);
observationInfo.Name = 'observations';
steerMax = pi/4;
discreteSteerAngles = -steerMax : deg2rad(15) : steerMax;
actionInfo = rlFiniteSetSpec(num2cell(discreteSteerAngles));
actionInfo.Name = 'actions';
numActions = numel(actionInfo.Elements);
% Create the Simulink environment interface to specify the path of the RL proxy block
blk = [mdl '/RL Controller/RL Agent'];
env = rlSimulinkEnv(mdl,blk,observationInfo,actionInfo);
%Specifies the reset function for training. This function resets the self-carrier's initial posture
env.ResetFcn = @autoParkingValetResetFcn;
```

进一步本文设置随机种子生成器以实现可重复性。并创建一个具有 16 个输入和一个输出的深度神经网络。批评网络的输出是特定观测值的状态值函数。使用并创建 PPO 代理的评价体系，再执行组件深度神经网络。最后指定代理选项并创建 PPO 代理。

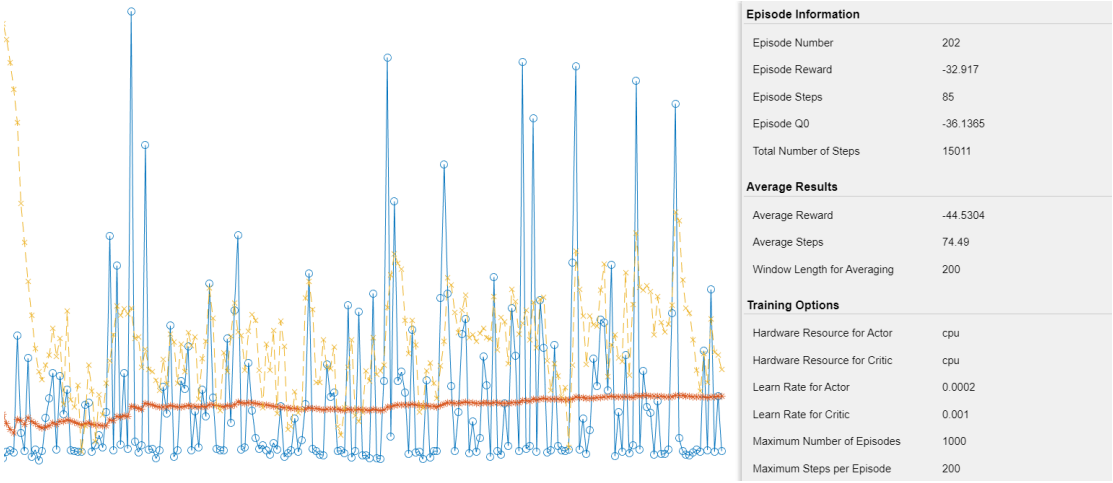
```
%Set the random seed generator for reproducibility
rng(0)
%create a deep neural network with 16 inputs and one output. The output of the critic
criticNetwork = [
    featureInputLayer(numObservations,'Normalization','none','Name','observations')
    fullyConnectedLayer(128,'Name','fc1')
    reluLayer('Name','relu1')
    fullyConnectedLayer(128,'Name','fc2')
    reluLayer('Name','relu2')
    fullyConnectedLayer(128,'Name','fc3')
    reluLayer('Name','relu3')
    fullyConnectedLayer(1,'Name','fc4')];
%Create the critic for the PPO agent
criticOptions = rlRepresentationOptions('LearnRate',1e-3,'GradientThreshold',1);
critic = rlValueRepresentation(criticNetwork,observationInfo,...
    'Observation',{'observations'},criticOptions);
%Create the actor deep neural network
actorNetwork = [
    featureInputLayer(numObservations,'Normalization','none','Name','observations')
    fullyConnectedLayer(128,'Name','fc1')
    reluLayer('Name','relu1')
    fullyConnectedLayer(128,'Name','fc2')
    reluLayer('Name','relu2')
    fullyConnectedLayer(numActions, 'Name', 'out')
    softmaxLayer('Name','actionProb')];
%Create a discrete categorical actor for the PPO agent
actorOptions = rlRepresentationOptions('LearnRate',2e-4,'GradientThreshold',1);
actor = rlStochasticActorRepresentation(actorNetwork,observationInfo,actionInfo,...
    'Observation',{'observations'},actorOptions);
```

最后只需要指定使用对象进行训练的选项，使用训练函数训练代理。但是训练此代理是一个计算密集型过程，需要几分钟才能完成，本文在保证质量的情况下把训练量调整到原来的十分之一，即 1000 个情景，可以保证在五分钟内完成训练。



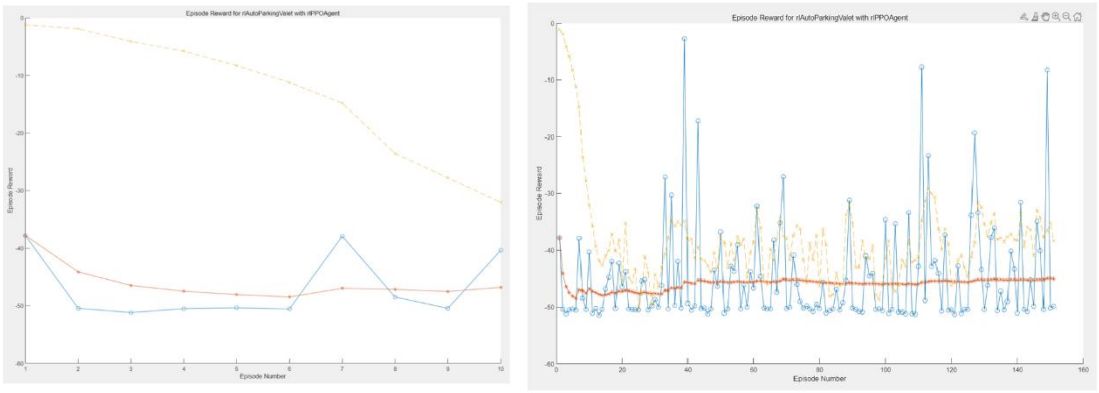
四、训练情况及效果展示

训练过程如上所示，小车会在选定位置中不断进行倒车尝试。下图所示为训练过程中的训练代理的实时效果图，其训练情况及其拟合效果会实时一并显示：

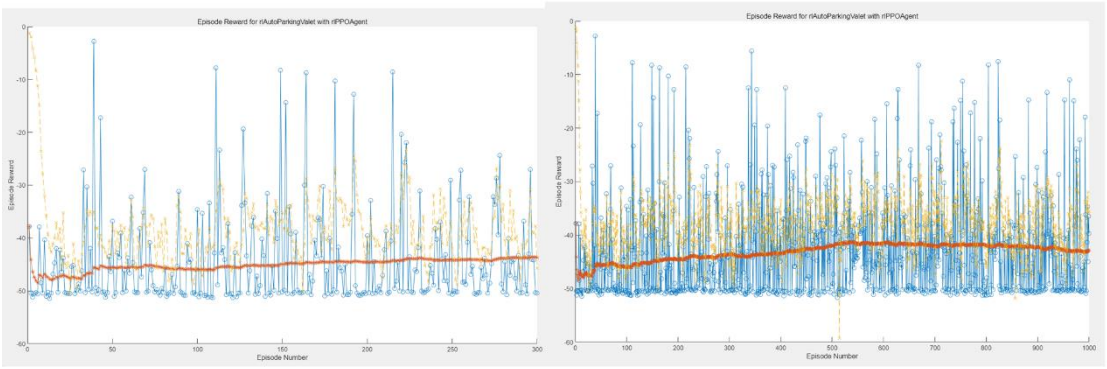


拟合表右侧分别展示的是 Episode Information, Average Results, Training Options, Final Results。 其中 Episode Information 包括 Number， Reward， Steps， Q0， Total Number of Steps 等； Average Results 包括 Rewards， Steps， Window Length for Averaging 等； Training Option 包括 Resource for Actor， Resource for Critic， Number of Episodes 等； Final Results 包括 Training Stopped by， Training Stopped at Value，Elapse Time 等。

一般情况下，训练效果越好，后期的拟合情况波动点会减少，趋势会更加平稳。

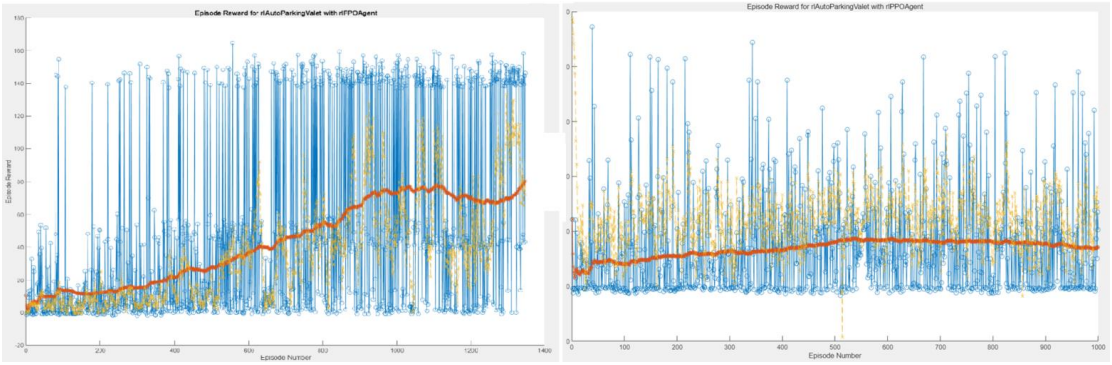


（训练 10 次和 150 次的拟合情况）



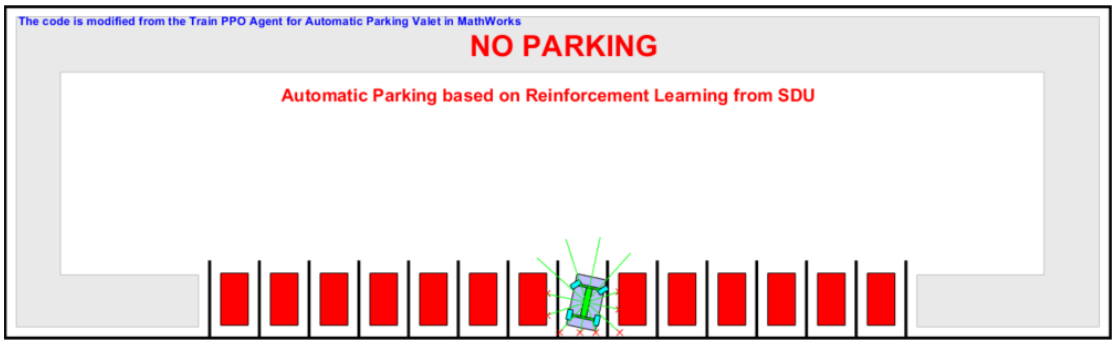
（训练 300 次和 1000 次的拟合情况）

本文将现有的拟合情况与 MathWorks 中的拟合情况进行对比，下图中右侧为本项目情况，左侧为 MathWorks 中情况：



可以清晰的发现，本项目的拟合情况不管在 Episode Information, Average Results, Training Options, Final Results 上都具有更加稳定的表现，可以有较强说服力的说明本项目的训练水平更好。

本文以停入第八个车位为目标进行测验，测试下小车可以自主且正常匀速地找到车位并倒入库中。测试动画切片如下所示：



以上为项目的整体设计过程，在 GitHub 文档中本文会附上演示视频，更多视频信息可以在发布在 Bili Bili 上的展示视频中观看。

五、参考网址

【1】2022-01-22，里程计模型（2）：阿克曼结构底盘， https://blog.csdn.net/weixin_45929038/article/details/122632369

【2】 2020-12-3，基于强化学习的自主泊车 MATLAB 实例，https://zhuanlan.zhihu.com/p/341155817

【3】MathWorks， Train PPO Agent for Automatic Parking Valet， Train PPO Agent for Automatic Parking Valet – MAT+LAB & Simulink (mathworks.com)