Robust Auto-parking: Reinforcement Learning based Real-time Planning Approach with Domain Template

Anonymous Author(s)
Affiliation
Address
`email`

Abstract

This paper presents an automatic parking for a passenger vehicle, with highlights on a robust real-time planning approach and on experimental results. We propose a framework that leverages the strength of learning-based approaches for robustness to environments noise and capability of dealing with challenging tasks, and rule-based approaches for its versatility of handling normal tasks, by integrating simple rules with RL under a multi-stage architecture, which is inspired by typical auto-parking template. By taking temporal information into consideration with using Long Short Term Memory (LSTM) network, our approach could facilitate to learn a robust and humanoid parking strategy efficiently. We present preliminary results in a high-fidelity simulator to show our approach can outperform a geometric planning baseline in the robustness to environment noise and efficiency of planning.

1 Introduction & Related Works

Automatic parking is an autonomous car-maneuvering system that moves a vehicle from a traffic lane into a parking spot to perform parallel, perpendicular or angle parking. The key idea is to plan and parameterize the basic control profiles of steering angle and speed, in order to achieve the desired shape of the vehicle's path within the available space and aim to enhance the comfort and safety of driving in constrained environments where much attention and experience are required to steer the car.

In automatic parking, our vehicle should be able to decide which actions to carry out both deliberatively and reactively w.r.t its goal and current situation while taking into account events in a timely manner [1]. There have been several approaches to generate the sequence of controlled motions for the parking problem, which could be generally categorized into two categories: rule-based approaches and learning-based approaches. The rule-based approaches, i.e. geometric planning that based on admissible circular arcs uses trajectories with easy geometrical equations, have strong ability for generalization, however, these approaches could only cover limited scenarios since they could not guarantee to provide drivable trajectories in complex situations; the learning-based approaches, i.e. using fuzzy logic or neural network to learn a human technique, can be effective solutions to deal with uncertainties and inaccuracies in the mapping of the environments, however, they can be limited to human experts' knowledge and difficult to generalize [2].

Reinforcement learning (RL) is a general framework for decision making, which optimizes the long-term accumulated rewards through interacting with the environment step by step. Thus, RL could deal with a task like automatic parking in natural. With the recent success of deep RL, in this work we propose a framework that leverages the strength of learning-based approaches for robustness to environments noise and capability of dealing with challenging tasks, and rule-based approaches for its versatility of handling normal tasks, by integrating simple rules with RL under a multi-stage architecture, which is inspired by typical auto-parking template that is universally used in geometric planning approach. Furthermore, we employ a recurrent neural network (RNN) to represent policy in RL and show that it could facilitate the policy network to have an ability to learn a robust and humanoid parking strategy under our proposed framework. We present preliminary benchmarks and show our approach can outperform a geometric planning baseline in the robustness to environment noise and efficiency of planning in a high-fidelity simulator.

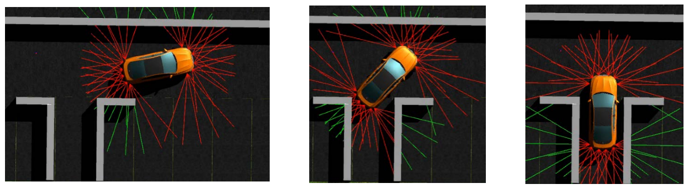
(a) Stage One (b) Stage Two (c) Stage Three

Figure 1: (a) For preparation. (b) For making full use of limited drivable space. (c) For fine-tune.

2 Framework

As one representative case, we only take perpendicular parking as an example to verify our approach. The functions of automatic parking could be roughly divided into two procedures: parking space and ego-vehicle location detection, trajectory planning and execution of maneuvers. For the first part, we adopt a common solution that to make the vehicle travel next to the target parking spot and scan it for its length and width with ultrasonic sensors, then utilize this information to build a local coordinate frame in order to locate ego-vehicle and every other boundary or static obstacles in parking space. For the second function, we treat path planning and execution of maneuvers as one single procedure. The ego vehicle is tasked with reaching the target parking spot from different starting positions with the minimum overhead of time while respecting the limits of speed and constraints from the vehicle and parking environment. The performance of the parking strategy is evaluated on its robustness in environment noise and trajectory smoothness with comparing to a geometric planning in a high-fidelity simulator.

2.1 Multi-stage Setting

Model-free deep RL's training efficiency is notorious low, especially with sparse rewards. In order to obtain more compact feedback signals, we decompose the whole parking task into three stages with inspiration from the typical auto-parking template shown in Figure 1. The first stage (Figure 1(a)) is for preparation with a subgoal to drive ego-vehicle to an ideal start pose for the next stage pose adjustment. While w.r.t pose adjustment, the subgoal for the second stage (Figure 1(b)) is to make full use of the parking space to adjust ego-vehicle's pose to target parking spot's pose by executing one or several back-and-forth shuttling maneuvers. The final stage (Figure 1(c)) is responsible for adjusting ego-vehicle to an ideal final parking pose with fine-tuning.

According to this multi-stage setting, we implement simple rules to achieve the second subgoal since the stage's function is relatively decoupled from the other two and the relationship between parking maneuvers and constraints from parking space is also straightforward. For stage one and three, we train the ego-vehicle with deep RL to obtain a skill of driving from different positions with various orientations to a target pose while considering the movement smoothness and flexibility, as well as the constraints from both ego-vehicle and parking space. We focus on learning lateral controls with a fixed idling longitudinal velocity since automatic parking is a typical low-speed scenario, which also facilitates the use of a discrete action space. Thus, we customized the steering angle with backward into 21 dimensions, which range from -540° to +540°.

The inputs of the value/policy network consist of both global and local state information. The global state information includes ego-vehicle's position and orientation in a coordinate frame that is deduced according to target parking spot (Figure 2(a)), while the local state information represents the relationship of ego vehicle with surrounding obstacles is collected from 12 ultrasonic sensors. Unlike the straightforward relationship between maneuvers and constraints in stage two, that is each forward or backward maneuver can be executed before hit any boundary while the numbers of maneuvers monotonically decrease the orientation difference between ego vehicle's current and target pose, the relationship between the current state and goal is nonlinear in the other two stages. Consequently, instead of designing continuous rewards we use following sparse reward function,

$$R = \begin{cases} -10, \quad \text{if collision or offset} \\ -0.05 * |\Theta_t - \Theta_{t-1}|, \quad \text{if large steering} \\ 10, \quad \text{if reach target area} \\ 0, \quad \text{otherwise} \end{cases} \tag{1}$$

where Θ denotes for ego-vehicle's steering angle. A negative terminal reward is given for failure (collision or offset) and a positive terminal reward is given for success (reaching the target area). A negative reward is given for large steering angle that denotes for $|\Theta_t - \Theta_{t-1}| > 54°$ which equals to a change of front wheel angle larger than 3.5°.

2.2 Policy Representation and Optimization

Deep Q Network (DQN) [3] based approach can learn the parking strategy successfully for the case with fixed initial state, however, it can not adapt to other initial parking states well. To explore a more efficient RL algorithm for stage one and three, adding human knowledge is the most intuitive approach. The approach we

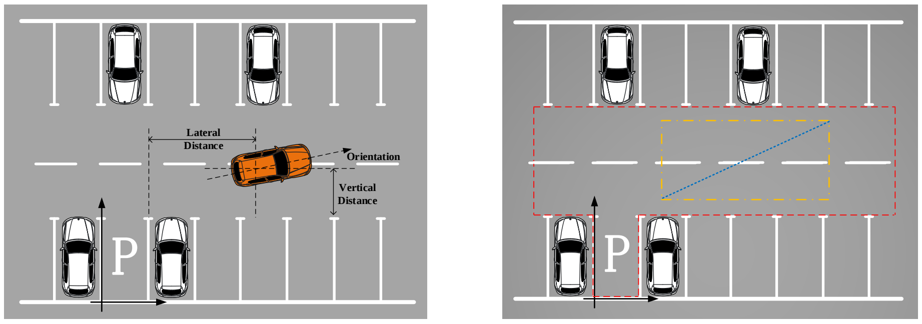

Figure 2: (a) Typical perpendicular parking environment with a coordinate frame conditional on target parking spot's position, orientation, and scenarios-related constraints/parameters (e.g. aisle width, parking spot size and obstacles distribution). (b) Assumption about task distribution for each phase (e.g. blue dotted line, yellow dash line with different ranges for corresponding orientations) for curriculum learning.

first investigate here is called Deep Q-learning from Demonstration (DQFD) [4]. This paradigm significantly accelerates the training but is not able to increase each DQN agent's capability of learning a generalized parking strategy, meanwhile collecting and preprocessing the demonstration data is time-consuming.

According to the performance of the implemented value-based RL algorithm, we infer that we might have to give credits to the temporal information contained in generated trajectories due to the strong causal relationship between the start and terminal parking states, which means the future states of the parking process not only depend upon on current state but also on the sequence of events that preceded it. Such property makes the parking problem a partially observable Markov decision process(POMDP) and thus non-Markovian from the viewpoint of our agent. The goal of dealing with a partial observed and non-Markovian RL problem is most likely beyond the abilities of traditional value function approach since it requires policies with an internal state, or memory, which can be the form of a trace of past observation/action pairs [5]. This motivates us to utilize RNN since it naturally provides a framework for dealing with policy learning using hidden state. Moreover, RNNs can be trained well with using gradient thus they are suited for policy gradient methods, which is one of the most popular policy-based RL algorithms [5].

We represent the policy as a general RNN. For each step, the policy receives a tuple (s, a, r, d) as input that embedded with using a function $\phi(s, a, r, d)$. The tuple consists of state s, action a, reward r and termination flag d that equals to 1 if the episode has terminated otherwise 0. In order to alleviate the vanishing and exploding gradients problem from RNNs' training, we use LSTM which has been demonstrated to have good empirical performance. The output of the LSTM is fed to a fully connected layer followed by a softmax function, which forms the categorical distribution over discrete actions.

Compare to value-based RL algorithms, the primary advantage of policy-based approaches is that they directly optimize the quantity of interest while remaining stable under function approximation with sufficiently small learning rate [6]. However, one of the two drawbacks is sample inefficiency because of that gradient estimation from high variance rollouts while the other is that they are extremely sensitive to the choice of learning rate. With regards to the pros and cons, the family of policy gradient methods we adopt in this work is named proximal policy optimization (PPO) algorithm [7] because of its simplicity of implementation and better sample complexity while maintaining some of the benefits of trust region policy optimization (TRPO) [8]. To reduce variance in the stochastic gradient estimation, we use a baseline function represented with a double-layer Multi-layer Perceptron (MLP).

2.3 Training Procedure and Curriculum Learning

We consider a set of tasks $D(T)$, where each is a parking task from a different initial state to the target terminal state under certain constraints from ego-vehicle, parking space, and static obstacles distribution. We denote each task by a tuple:

$$T = (L_T, P_T(s), P_T(s_{t+1}|s_t, a_t), H) \tag{2}$$

With using a policy-based approach, we need to optimize a stochastic policy $\pi_\theta : S \times A \rightarrow \mathbb{R}_+$ that parameterized by θ. Given the state $s_t \in P_T(s)$ that perceived at time t for task T, our agent predicts a distribution of actions with the policy, from which an action $a_t \sim \pi_\theta(a_t|s_t)$ is sampled. The agent interacts with the environment and perceives next state $s_{t+1} \sim P(s_{t+1}|s_t, a_t)$ and the immediate reward R_t according to the reward function defined in Eqn.(1). Thus, $P_T(s)$ and $P_T(s_{t+1}|s_t, a_t)$ define the environment in task T. L_T is a loss function that maps a trajectory τ collected from task T to a loss value, where τ=$(s_0, a_0, R_0, ..., s_H, a_H, R_H)$ with H denotes for the horizon. The loss of a trajectory is a negative cumulative reward $L_T(\tau) = -\sum_{t=0}^{H} R_t$.

Our goal is to find a parking strategy that is able to accomplish the parking from various initial states which is given access to a limited experience on each task from $D(T)$. The policy update procedure is shown in Figure 3. For each task, collect N trajectories under policy π_θ that denoted by $\tau_\theta^{1:N}$. We then calculate the gradient of

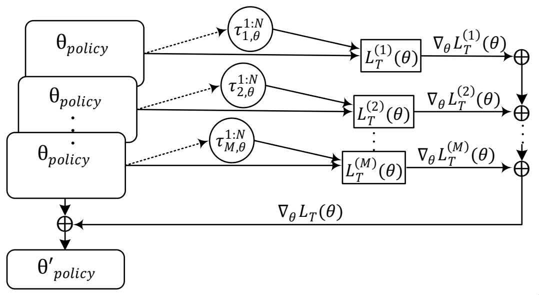

Figure 3: Computation graph for the update from θ_{policy} to θ'_{policy}.

$L(\tau_\theta^{1:N})$ w.r.t to θ, denoted $\nabla_\theta L(\tau_\theta^{1:N})$ where,

$$L_T(\tau_\theta^{1:N}) = \frac{1}{k}\sum_{n=1}^{N} L(\tau_\theta^n) \quad \text{and} \quad \tau_\theta^n \sim P_T(\tau|\theta) \tag{3}$$

We optimize θ by minimizing the expected loss over the distribution of task $D(T)$,

$$\min_\theta E_{T\sim D(T)}[L_T(\theta)] \quad \text{where} \quad L_T(\theta) = L_T(\tau_\theta^{1:N}) \tag{4}$$

Thus we collect N trajectories from M different tasks simultaneously and calculate the gradient w.r.t to each task, denoted $\nabla_\theta L_T^{(1)}(\tau_{1,\theta}^{1:N}), ..., \nabla_\theta L_T^{(M)}(\tau_{M,\theta}^{1:N})$. Consequently, we update the parameters of the policy using these gradient w.r.t θ:

$$\theta := \theta - \alpha\frac{1}{M}\sum_{i=1}^{M}\nabla_\theta L_T^{(i)}(\tau_{i,\theta}^{1:N}) \tag{5}$$

In order to enable our agent to obtain a more generalized driving skill w.r.t larger range of initial states, we employ curriculum learning that could help intelligent agents to learn better with organized examples which gradually illustrate more and more complex concepts[9]. For instance, in stage one we separate the training process into multiple phases and make different assumptions about the distribution of tasks for each training phase as shown in Figure 2(b). The initial positions are first drawn uniformly from a distribution $D_0(T)$ that could be described by the dotted line in Figure 2(b) with orientations randomly selected from $[-\pi/24, \pi/24]$. After K_0 iterations, we extend the task distribution from the dotted line to the dash-dot rectangle with orientations bounded by the same range denoted by $D_1(T)$. Consequently, to iterate another K_1 our agent is able to park from a certain area with orientations bounded by a small range. Furthermore, in order to increase the coverage of the initial orientation, we customize the task distribution $D_2(T)$ for the last training phase with tailored orientations that ranges in $[-\pi/8, -\pi/12]$, $[-\pi/12, -\pi/24]$, $[\pi/24, \pi/12]$, and $[\pi/12, \pi/8]$ w.r.t areas that guarantee no collision at the very beginnings. The corresponding training procedure is shown in Algorithm 1.

Algorithm 1 Training of Curriculum Learning

```
Input: training tasks $D_0(T)$, $D_1(T)$, $D_2(T)$, numbers of iterations $K_0$, $K_1$, $K_2$,
       batch size $M$ of tasks for each update, number of trajectories $N$ to collect for each task
Output: policy $\theta$
 1: Randomly initialize $\theta$
 2: while not done do
 3:   for $(K, D(T)) \in \{(K_0, D_0(T)), (K_1, D_1(T)), (K_1, D_2(T))\}$ do
 4:     for $k \in \{1, \dots, K\}$ do
 5:       Sample a batch of tasks $T \in D(T)$ with a batch size equals to $M$
 6:       for all $T$ do
 7:         Sample $N$ trajectories $\tau^{1:N}$ with policy $\theta$
 8:         Compute the gradient of $\theta$ w.r.t loss function as given in Eqn.(4)
 9:       end for
10:       Perform update $\theta := \theta - \alpha\frac{1}{M}\sum_{i=1}^{M}\nabla_\theta L_T^{(i)}(\tau_{i,\theta}^{1:N})$
11:     end for
12:   end for
13: end while
```

3 Results

To evaluate the performance of our approach, we compare it against two baselines that based on geometric planning and DQFD, and a MLP policy learned under our approach framework. All experiments are done in a high-fidelity simulator. For such benchmarks we set the scenario-related parameters as follows: parking spot size equals to $6.0 \times 2.5m$ with aisle width is larger than $6m$, ego-vehicle size is $4.93 \times 1.87m$, and the distance from ego-vehicle's back axle center to tail-stock is $1.04m$.

Table 1: Benchmark results for robustness to the initial state of our approach against baselines and MLP policy learned under our approach framework. The geometric meaning of column names are represented in Figure 2(a).

	Ours		**Baselines**	
	LSTM(PPO)	MLP(PPO)	Geometric Planning	DQFD
Vertical Distance (m)	$[1.5, 3.5]$	$[2.0, 3.0]$	$[2.0, 3.5]$	$[2.5, 3.0]$
Lateral Distance(LD) (m)	$[1.0, 12.0]$	$[2.0, 6.0]$	$[1.0, 12.0]$	$[2.0, 3.0]$
Orientation (°)	$[-10, 45]$, LD ≤ 1.5	$[-15, 15]$	$[-10, 45]$, LD ≤ 1.5	$[-5, 5]$
	$[-25, 30]$, LD > 1.5		$[-5, 30]$, LD > 1.5	

Robustness to Environment Noise: We explored the robustness of the learned policy with our approach to the changes in the environment noise, which infer the noise in the initial state and the detection of the target parking spot. We first measure the boundary of initial state that each approach is able to cover. The comparison results are summarized in Table 1. Generally, the geometric planning baseline covers a smaller vertical distance range, which is $0.5m$ less than the policy (LSTM) learned by our approach. Under this premise, the range of orientation that geometric planning baseline could cover is identical to our approach with a restricted lateral distance range $[1.0, 1.5]m$, however, 20° less when the lateral distance exceeds $1.5m$. Also, the ranges of the initial states that the policies learned with DQFD and our approach as represented by MLP could cover are quite limited. Thus, the policy (LSTM) learned by our approach is the most robust to the noise in the initial state.

We further explore the policy's robustness to the noise in the target parking spot's detection by fixing other scenario-related parameters. As described in Section 3.2, ego-vehicle's position and orientation are deduced according to the target parking spot, consequently adding noise with a certain probability constantly to the target spot detection will randomly add or subtract a value from ego-vehicle's position at each step (e.g.10Hz). Thus, we measure the success rate of the policy in such randomly perturbed environment, the results are summarized in Table 2. It shows that with adding a lateral noise less or equals to $0.3 * 2.5m$ and a vertical noise less or equals to $0.3m$ either independently or simultaneously, the policy (LSTM) learned with our approach could still remain a 100% success rate out of 200 test trials. Compared with the geometric planning baseline, which is extreme inefficient or even not available for re-planning w.r.t changes in the parking spot's position detection during the parking process, our learned policy is much more robust to such noise since while the action at each step may change but the subgoal of each stage remains the same thus real-time high frequency (e.g. per step) re-planning is allowed, which is a typical human-like driving behavior.

Also, we found that both under our approach framework, the learned policy represented by MLP is much less robust to the initial state than the LSTM learned policy (Table 1). Figure 4 illustrates the trajectory comparison between these two policies. An obvious parking pattern for MLP policy is that it tends to first drive to a same point in the first stage no matter what initial state it starts with, thus the trajectories for stage two and three looks almost identical. Although we let the policy investigate different tasks simultaneously during the training, MLP is more likely learned an average parking strategy based on shorter-term since its lack of hidden state structure. Consequently, such property limits its ability to learn a parking strategy that is more robust to the initial state.

Moreover, in comparison to the geometric planning baseline, we found that our approach is beneficial when the task is challenging (e.g. narrow aisle and tiny parking spot). The success rate of our learned policy (LSTM) could remain 100% when aisle width falls in the range [*Ego-vehicle length*, $6m$] with parking spot's width is at least $2.5m$, while the geometric planning baselines' probability of success parking significantly drops.

Trajectory Smoothness: Finally, we illustrate the smoothness of the trajectories. We collect two sets of parking trajectories generated by the learned parking strategy (LSTM) with our approach and geometric planning

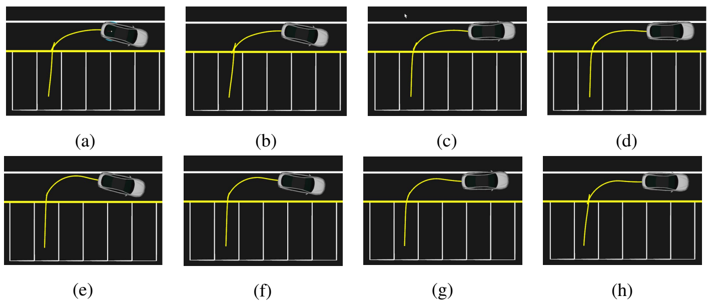

Figure 4: (a)-(d) Trajectories generated by LSTM policy under our approach. (e)-(h) Trajectories generated by MLP policy under our approach. (a)(e),(b)(f),(c)(g) and (d)(h) pairs each have same initial state.

Table 2: Benchmark results for robustness to environment noise of our approach, with listing lateral and vertical noise that added to target parking spot in terms of σ_x and σ_y, $\sigma_x = \alpha \times$ parking spot width.

Noise in Target Spot Detection (m)	**Success Rate of Parking (%)**
$\sigma_x \leq 0.3 * 2.5$	100.0
$0.3 * 2.5 < \sigma_x \leq 0.5 * 2.5$	83.5
$0.5 * 2.5 < \sigma_x \leq 0.8 * 2.5$	72.5
$\sigma_y \leq 0.3$	100.0
$0.3 < \sigma_y \leq 0.4$	81.0
$0.4 < \sigma_y \leq 0.5$	49.0
$\sigma_x \leq 0.3 * 2.5, \sigma_y \leq 0.3$	100.0
$0.3 * 2.5 < \sigma_x \leq 0.5 * 2.5, \sigma_y \leq 0.3$	69.0
$0.5 * 2.5 < \sigma_x \leq 0.8 * 2.5, \sigma_y \leq 0.3$	31.5
$\sigma_x \leq 0.3 * 2.5, 0.3 < \sigma_y \leq 0.4$	79.5
$0.3 * 2.5 < \sigma_x \leq 0.5 * 2.5, 0.3 < \sigma_y \leq 0.4$	48.5
$0.5 * 2.5 < \sigma_x \leq 0.8 * 2.5, 0.3 < \sigma_y \leq 0.4$	26.0
$\sigma_x \leq 0.3 * 2.5, 0.4 < \sigma_y \leq 0.5$	53.5
$0.3 * 2.5 < \sigma_x \leq 0.5 * 2.5, 0.4 < \sigma_y \leq 0.5$	28.0
$0.5 * 2.5 < \sigma_x \leq 0.8 * 2.5, 0.4 < \sigma_y \leq 0.5$	11.5

respectively with success parking and limited time cost as premises. We calculate the smoothness for each trajectory with the following function,

$$\frac{\sqrt{\textbf{Var}(\tau_a)}}{\mid \textbf{Avg}(\textbf{Diff}(\tau_a)) \mid} \quad \text{with} \quad \textbf{Diff}(\tau_a) = \sum_{i=0}^{H-1} (a_{i+1} - a_i) \tag{6}$$

where τ_a denotes for a sequence of output steering commands, a_i is the output for i^{th} step, and H is the trajectory length. The average smoothness of the trajectories collected by our approach is approximately 34.11 which is much smaller than that of the geometric planning trajectories, which is approximately 163.47. We found that our learned parking strategy is less tend to sharply turn the steering wheel to the very left/right, and don't need to steer at stops as what geometric planning approach normal would do, which can wear the tires and heat the electric power steering. Comparison of the trajectories generated by these two approaches is illustrated in Figure 5.

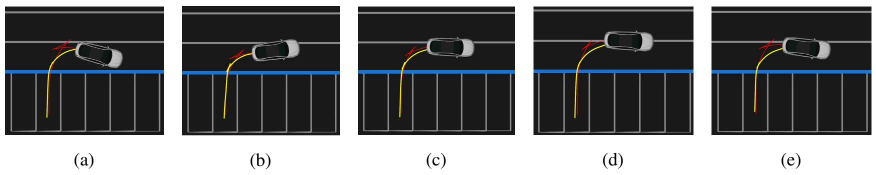

Figure 5: (a)-(d) Illustrate trajectory comparison between our approach with LSTM policy and geometric planning approach, yellow trajectory denotes for our approach and red trajectory denotes for geometric planning.

4 Conclusion and Future Work

We proposed a framework that leverages the strength of learning-based approaches for robustness to environments noise and capability of dealing with challenging task, and rule-based approaches for its versatility of handling normal tasks, by integrating simple rules with RL under a multi-stage architecture, which inspired by the domain template. Such a framework provides a mechanism for us to incorporate prior knowledge for decomposing a task into a multi-stage problem with shorter-term rewards, which can lead to fast convergence to successful parking policies. Also, by taking temporal information into consideration by using LSTM for policy representation, our approach could facilitate to learn a more robust and humanoid parking strategy efficiently. We present preliminary benchmarks and show our approach can outperform a geometric planning baseline in a high-fidelity simulator in the robustness to environment noise and efficiency of planning.

Our work represent that RL could solving parking problem under static environment efficiently while showing the ability for generalization. In future, we foresee using model-based RL with optimization-based planning approach and model-free RL to accomplish parking problem under more complex and dynamic scenarios.

References

[1] R. Alami, R. CHatila & S. Fleury (1998) An Architecture for Autonomy. *The International Journal of Robotics Research* **17**(4):315-337

[2] Hélène Vorobieva, Sébastien Glaser, Nicoleta Minoiu-Enache & Saïd Mammar (2015) Automatic parallel parking in tiny spots: path planning and control. *IEEE Transactions on Intelligent Transportation Systems* **16**(1)396-410

[3] Volodymyr Mnih, Koray Kavukcuoglu, David Silver & Andrei A. Rusu (2015) Human-level control through deep reinforcement learning. *Nature* **518**, 529-533

[4] Todd Hester, Matej Vecerik, Olivier Pietquin & Marc Lanctot (2018) Deep Q-learning from demonstrations. *AAAI*

[5] Daan Wierstra, Alexander Förster, Jan Peters & Jürgen Schmidhuber (2010) Recurrent policy gradients. *Logic Journal of the IGPL* **18**(5)620-634

[6] Ofir Nachum, Mohammad Norouzi, Kelvin Xu & Dale Schuurmans (2017) Bridging the gap between value and policy based reinforcement learning. *NIPS*

[7] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford & Oleg Klimov (2017) Proximal policy optimization. *arXiv preprint arXiv:1707.06347*

[8] John Schulman, Sergey Levine, Philipp Moritz, Michael Jordan & Pieter Abbeel (2015) Trust region policy optimization. *CoRR, abs/1502.05477*

[9] Yoshua Bengio, Jerome Louradour, Ronan Collobert & Jason Weston (2009) Curriculum learning. *ICML*