Highway Traffic Modeling and Decision Making for Autonomous Vehicle Using Reinforcement Learning

Changxi You Jianbo Lu Dimitar Filev Panagiotis Tsiotras

***Abstract*— This paper studies the decision making problem of autonomous vehicles in traffic. We model the interaction between an autonomous vehicle and the environment as a stochastic Markov decision process (MDP) and consider the driving style of an experienced driver as the target to be learned. The road geometry is taken into consideration in the MDP model in order to incorporate more diverse driving styles. By designing the reward function of the MDP, the desired, driving behavior of the autonomous vehicle is obtained using reinforcement learning. Simulated results demonstrate the desired driving behaviors of an autonomous vehicle.**

Keywords: Reinforcement learning, decision making, autonomous vehicle, Markov decision process.

I. Introduction

Autonomous vehicles are expected to significantly improve traffic congestion, reduce collisions and resulting injuries, enhance mobility for the elderly and the disabled, and reduce the need for parking space in cities [1]. The planning strategy of an autonomous vehicle consists of three tasks, which include: *mission planning*, where the vehicle solves a routing problem in order to complete a task; *decision making*, where the vehicle chooses an appropriate action for the next time step from an available action set; and *path planning*, where the vehicle plans its future trajectory as a function of space or time [2]–[5]. This paper focuses on the problem of decision making for autonomous vehicles in traffic. Specifically, we wish to reproduce the decision making of an expert driver, that is, we wish to duplicate the optimal driving strategy involving several typical driver actions such as lane-shifting, lane and speed maintaining, accelerating and braking, by also considering the stochastic driving behaviors of the other vehicles in traffic.

One of the first autonomous vehicle was developed by Carnegie Mellon University's Navlab in 1988, and it was able to achieve lane-following using camera images [6]. Navlab completed the first autonomous coast-to-coast trip across the United States in 1995, traveling 2,849 miles between Pittsburgh and San Diego at an average speed of 63.8 mph [7]. Another important milestone in the self-driving vehicle technology was the DARPA Grand Challenge, which was held three times between 2004 and 2007 [8]. In these races the vehicles were required to drive autonomously in an off-road course (2004 and 2005) or an urban area course (2007) without any human intervention. Those tests showed that fully autonomous off-road driving and fully autonomous urban driving are technologically possible. Since then, many commercial companies, startups, and research organizations have launched their own development of autonomous vehicles.

In order to generate a smooth path for an autonomous vehicle, Choi et al. [9], [10] presented a series of path planning algorithms based on Bézier curves. The planned paths have continuous curvature and satisfy the road boundary constraints. Ulbrich and Maurer [11] used a partial observable Markov decision process (POMDP) to model the decision making for lane changes, and implemented a two-step algorithm in real-time to obtain the optimal action for an autonomous vehicle in an urban driving task. Kuwata et al. [4] proposed a real-time path planning algorithm based on Rapidly-exploring Random Trees (RRTs). This algorithm was implemented on an autonomous vehicle which completed a 60 mile simulated military supply mission in the 2007 DARPA Urban Challenge. A more extensive survey on path planning for autonomous vehicles can be found in [8], [12].

Other techniques using ideas from artificial intelligence (AI) have also been developed to solve planning problems. These include supervised learning [13], deep learning [14] and reinforcement learning [15]. Lange et al. [16] used a deep neural encoder to extract feature representations from the raw visual input of camera images for a racing vehicle, and successfully learned the optimal control actions (i.e., steering, accelerating and braking) using reinforcement learning. The control performance was even better than an experienced human driver. The approach in [16] concentrated on improving the driving performance of a single vehicle without considering the traffic. Shalev-Schwartz et al. [2] took the traffic into consideration and divided the planning problem into two phases. They first modeled the state transition of the traffic using a deep neural network, such that they could apply supervised learning to predict the near future states of the system. Subsequently, they used a recurrent neural network to model the trajectory and learn the optimal driving policy of the autonomous vehicle. This approach does not rely on any Markovian assumption, and hence it is considered to be robust to the stochastic behavior of the environment. The learning procedure was validated using both an adaptive cruise control task and a roundabout merging task.

This paper proposes a new MDP model to represent the stochastic behaviors of the environmental vehicles in highway traffic. This model differs from previous similar

MDP models [17], [18] in the sense that we take the road geometry into consideration in order to compare and analyze different driving strategies during cornering. Another advantage is that the model is easily scalable to have more vehicles and more lanes in traffic. Unlike the MDP models of [19] and [18] that need to discretize the velocity of each vehicle in traffic (which leads to problems with a large state space), we remove the velocities of the vehicles from the MDP model and consider them only in the perception layer or in the control layer.

The rest of the paper is organized as follows: Section II introduces the traffic model using a stochastic MDP. Section III designs the reward function and solves the MDP problem using Q-learning, and Section IV implements the RL algorithm, and analyzes the results. Finally, Section V summarizes the results of this study.

II. Traffic Modeling

In this section we use a Markov decision process (MDP) [20] to model the interaction between the autonomous vehicle and the surrounding vehicles in traffic. In the following sections we use RL to solve this MDP problem for the optimal policy that achieves the desired driving behaviors. We start with a brief overview of MDPs.

A. Markov Decision Process

Markov decision processes (MDPs) are used in a wide area of applications such as robotics, economics, manufacturing and automatic control. An MDP is a mathematical framework that probabilistically models the interaction between an agent and the environment, pioneered by the work of Bellman [20]. The agent is assumed to be a learner or decision maker, who interacts with the environment [15]. It receives a reward and a representation of the environment's state at each time step, and exerts an action on the environment that may change its future state.

A typical MDP is represented using a 6-tuple $(S, A, \mathscr{T}, \gamma, D, R)$, where S is a (finite) set of possible states that represent a dynamic environment, A is a (finite) set of available actions that the agent can select at a certain state [1], $\mathscr{T}$ is the state transition probability matrix that provides the probability of the system transition between every pair of the states, $\gamma \in [0,1)$ is the discount rate that guarantees the convergence of total returns, D is the initial-state distribution, and R is the reward function that specifies the reward gained at a specific state by taking a certain action.

The core problem of an MDP is to find a policy π for the agent, where the policy $\pi : S \to A$ specifies the action to take at the current state s_t. The goal is to find the optimal policy π^* that maximizes the cumulative discounted reward over an infinite horizon:

$$\pi^* = \arg\max_{\pi} \mathbb{E}\Big[\sum_{t=0}^{\infty} \gamma^t R(s_t, \pi(s_t))\Big], \tag{1}$$

where the term $R(s_t, \pi(s_t))$ represents the reward the agent receives by taking an action determined by policy π at the present state s_t.

[1] Without loss of generality we will assume that all actions are available in each state.

B. System Modeling

The MDP to be used to model the traffic is based on the following observations. Consider a typical scenario of traffic in a multi-lane road, as shown in Figure 1. Each vehicle moves in the middle of each lane with the average speed of the traffic flow.

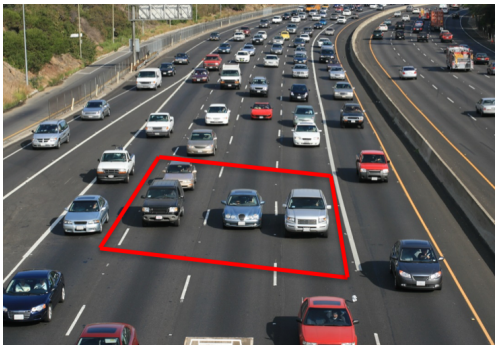

Fig. 1: The traffic on multi-lane road.

Let us now consider the driving behavior of the blue vehicle in the middle of the red rectangular shown in Figure 1. There are several actions the blue vehicle can take. For instance, it can maintain its current speed, accelerate or brake to occupy the vacant positions ahead of it or behind it, or move to the left or to the right lane if there is no chance for a collision. Assuming that each driver intends to maximize a certain reward function, if one can obtain the reward function of an "expert/experienced" driver, one should be able to reproduce this expert driver's behaviors using reinforcement learning techniques [15].

In the following, we designate the vehicle we want to control as the host vehicle (HV), and all remaining vehicles in traffic as the environmental vehicles (EVs). We assume that the drivers of different vehicles do not communicate with one another, and also that the vehicles do not share data with each other. Hence, the MDP system has only a single actively controlled agent. The available action set for each vehicle in traffic is given by $A \triangleq$ {"maintain", "accelerate", "brake","left-turn", "right-turn"}.

1) State Definition: By considering the positions of the HV, and the number and positions of the EVs around the HV, we define the state of the MDP as shown in Figure 2.

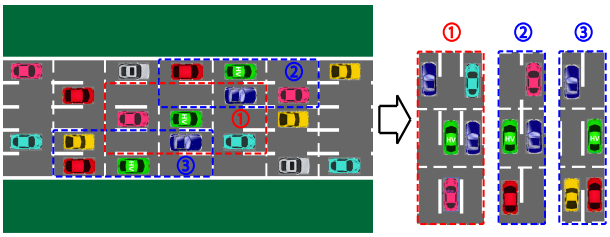

Fig. 2: The cells and the definition of the state: ① 9-cell internal-lane state, ② 6-cell left-boundary state and ③ 6-cell right-boundary state

In Figure 2, we use the white dashed lines to divide the road into small cells and use the green vehicle to denote the HV. The states of the MDP represent either of the three conditions shown in Figure 2: 1) the HV is in the middle lane of the road, where we use nine cells to represent the state, and 2) and 3) where the HV is next to the road boundaries and we use six cells to represent the current state. Taking all possible combinations into account, the number of the internal-lane states is $2^8 = 256$, and the number of the left(right)-boundary states is $2^5 = 32$. Hence the total number of the states of the MDP is $256 + 2 \times 32 = 320$. Note that the approach can be easily extended to highways with any number of lanes and vehicles [2].

Figure 3 shows a possible overtaking behavior of the HV (green car) during a left-turn corner. The HV driver may prefer overtaking the pink car in front from the left rather than from the right. In order to investigate the effect of the road geometry on the observed driving behaviors of different drivers, in this work we take the road curvature into account and consider three kinds of roads, namely, left-turn, right-turn, and straight roads. The total number of the states is therefore $320 \times 3 = 960$. It is worth mentioning that, although we only consider three kinds of road geometries, one can, similarly, divide the road characteristics into more classes, as needed. For instance, one could also take into account different slopes of the roads, such as downhill, uphill, flat roads etc.

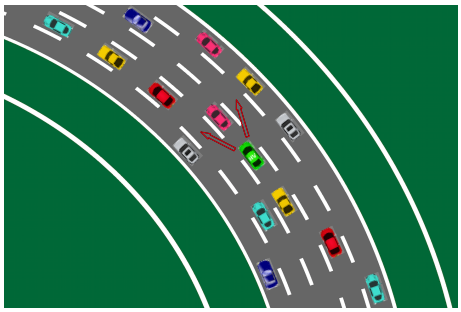

Fig. 3: Overtaking during cornering.

[2]The traffic model is also possible to be used for modeling urban traffic by adjusting the state definition. For instance, the cell size may be defined to change with the size of each EV and its velocity relative to the HV.

2) State Transitions: We want to model the state transition process by mimicking the traffic in real world scenarios. To this end, we make the following assumptions: 1) the number of lanes n is free and greater than equal to two ($n \geq 2$), 2) the number of EVs N is free but no larger than eight, given the cell geometry of Figure 2 ($0 \leq N \leq 8$), 3) the EVs have their own policies that may be different from the HV, 4) the EVs take a random action, 5) no collision arises from the actions of the EVs, and 6) each vehicle takes a single action at each time step.

The state transition procedure from the current state s_t to the next state s_{t+1} is given in two steps: First, the HV observes the current state s_t and selects an action $\pi(s_t)$ following its current policy. Second, the EVs respond to the action of the HV, and take an action following their own policies in a random sequence. The new positions of the HV and the EVs around the HV define the next state s_{t+1}. This state transition process is demonstrated in Figure 4.

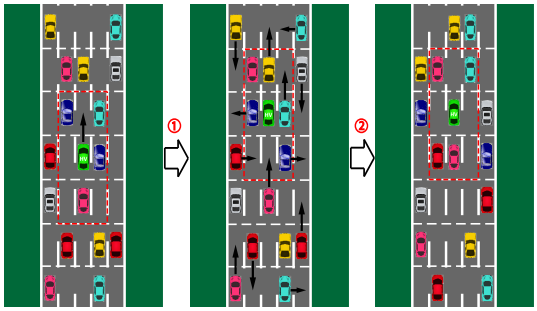

Fig. 4: State transition process.

The current state s_t is defined using the nine cells in the red rectangular on the left graph in Figure 4. Based on s_t, the HV may brake or switch to the right lane but these actions will result in an collision. The available safe actions of the HV are maintaining, accelerating and switching to the left lane. For instance, suppose that the HV accelerates and occupies the cell in front of it. As a consequence, the red rectangular also moves since the EVs surrounding the HV change. Next, all EVs respond to the action of the HV and take an action following a certain policy in a random order. The next state s_{t+1} is obtained after all vehicles complete their actions (see the red rectangular on the right graph in Figure 4).

III. Reinforcement Learning

In this section we design the reward function and use reinforcement learning techniques to solve the MDP problem to determine the optimal policy.

A. Reward Function

A widely used approach to design the reward function is to represent it as a function of some manually chosen features. These features depend on the action of the agent and the state of the environment. We use a linear

combination of the features to represent the reward function [21]–[25]:

$$R(s,a) = w^{\mathrm{T}}\Phi(s,a), \tag{2}$$

where w is the weight vector, and $\Phi(s,a)$ is the feature vector with each component representing a single feature point in the state-action space. Possible choices of feature points may be the binary values indicating whether a certain argument is true or not. In this work we define the features in $\Phi(s,a)$ as follows:

1) Action features. The driver may prefer taking certain actions than others if he receives a higher reward from these actions.

2) Position of the HV. It indicates whether the HV is driving next to the road boundaries. The driver may prefer to drive in different lanes, depending on the road geometry.

3) Overtaking strategy. This feature is used to achieve different overtaking behaviors during cornering. The driver may have a preference in regards to overtaking the front car either from the left or from the right.

4) Tailgating. The value of this feature is "true" if the HV is behind an EV and "false" otherwise.

5) Collision incident. Collision occurs if the HV and a EV appear in the same cell.

One can design the weight vector w to encourage or discourage certain features using the given reward function, and then use reinforcement learning to learn the corresponding optimal policy by maximizing the total reward. Another idea is to design the reward function using a parameterized function approximator such as a Gaussian process [24], [25] or a DNN [26]. The parameters of the function approximator are hard to design manually since they may not be directly related to features that have clear physical meaning, and hence they can only be learned from data. This approach is called inverse optimal control or inverse reinforcement learning [27]. Due to the lack of the space, in this paper we do not investigate further the inverse reinforcement learning approach for decision making.

B. Q-Learning

Q-learning is a typical model-free RL algorithm. It was first introduced in Watkins's PhD thesis in 1989 [28]. Different variants of Q-learning were developed to solve various reinforcement learning problems [29]–[31].

The Q-learning algorithm works directly on the state-action value $Q^{\pi}(s_t,a_t)$, which represents the expected discounted cumulative reward starting at state s_t, taking action a_t and then following policy π, afterwards. The update law of the Q values can be expressed as follows [15], [28],

$$Q(s_t,a_t) \leftarrow Q(s_t,a_t) + \alpha\Big(R_t + \gamma \max_{a_{t+1}\in A} Q(s_{t+1},a_{t+1}) - Q(s_t,a_t)\Big), \tag{3}$$

where $\alpha \in [0,1]$ is the learning rate (step size), which determines how much the newly acquired information overrides the current Q values, and $\gamma \in [0,1)$ is the discount rate, which describes the importance of future rewards for the agent.

The Q-learning algorithm is summarized in Algorithm 1.

Algorithm 1 Q-Learning Algorithm

Input: S, A, α, γ, ϵ, R
Output: Q^*, π^*

$Q \leftarrow Q_0$
$Q(s_{\text{final}},\cdot) \leftarrow 0$
Converge $\leftarrow$ False
while not Converge **do**
  $s \leftarrow s_0$
  EpisodeOver $\leftarrow$ False
  while not EpisodeOver **do**
    $a \leftarrow \max_{a\in A} Q(s,a)$ (i.e., ϵ-greedy)
    $s' \leftarrow$ state after taking action a
    if $s' \in s_{\text{final}}$ **then**
      EpisodeOver $\leftarrow$ True
    else
      $Q(s,a) \leftarrow Q(s,a) + \alpha\Big(R(s,a) + \gamma \max_{a\in A} Q(s',a) - Q(s,a)\Big)$
    $s \leftarrow s'$
  if Q converges **then**
    Converge $\leftarrow$ True
$Q^* \leftarrow Q$
$\pi^*(s) = \max_{a\in A} Q^*(s,a)$

IV. Results and Analysis

In this section we implement the previous RL algorithm on the proposed traffic model and analyze the results.

We show two different driving behaviors using RL, namely, overtaking and tailgating. To this end, we use the features defined in Section III and design the weights w_1 and w_2 to achieve the two desired driving behaviors, respectively. The weights are provided in Table I.

The desired driving behavior by designing w_1 is to show overtaking, which can be described as follows: 1) The HV accelerates to occupy the front cell if it is available; 2) The HV maintains its velocity if there is an EV in front of it and no overtaking is possible; 3) The HV overtakes the front EV if only one side is available for overtaking, by lane-shifting first and then accelerating and maintaining constant speed; 4) The HV overtakes the front EV from the inner side of the corner if both the left and right sides are available; 5) The HV does not change lane unless for overtaking; 6) The HV does not brake to occupy the rear cell. 7) No collision is allowed.

The desired driving behavior by designing w_2 is to demonstrate tailgating, which can be described as follows: 1) The HV maintains its velocity if there is an EV in front of it; 2) The HV accelerates to occupy the front cell if it is available and no tailgating occurs by changing lanes; 3) The HV changes lane to tailgate an EV if there is no EV in front of it; 4) The HV prefers to tailgate the vehicle in the lane closer to the inner curb of the road

TABLE I: The selected features and the weights for reinforcement learning.

$\Phi(s,a)$	w_1	Interpretation	w_2	Interpretation
maintain	0	NA	0	NA
accelerate	0.075	Prefer accelerating	0.05	Prefer accelerating
brake	-0.625	Avoid braking	-0.5	Avoid braking
left-turn	-0.05	Reduce lane-shifting	-0.025	Reduce lane-shifting
right-turn	-0.05	Reduce lane-shifting	-0.025	Reduce lane-shifting
HV position	0	NA	0	NA
overtake	0.05	Prefer inner overtaking	0.025	Prefer inner tailgating
tailgate	0	NA	0.225	Prefer tailgating
collision	-0.15	Avoid collision	-0.15	Avoid collision

in a corner; 5) The HV does not change lanes unless for tailgating; 6) The HV does not brake to occupy the rear cell; 7) No collision is allowed.

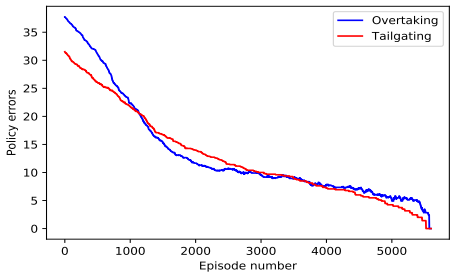

Fig. 5: The convergence performance of the policy π.

We implemented the Q-learning in Algorithm 1 to learn the optimal policies using both w_1 and w_2 with learning rate $\alpha = 0.75$, discount rate $\gamma = 0.5$, and $\epsilon = 0.08$ for the ϵ-greedy principle (Line 8 of Algorithm 1). Figure 5 shows the convergence of the policies π_1^* and π_2^* corresponding to w_1 and w_2, respectively. In both cases, it takes less than five minutes to obtain the results shown in Figure 5 on a dual-core 2.27 GHz Intel Xeon processor running 64-bit Windows 10 Enterprise operating system and programmed using Python 3.6.

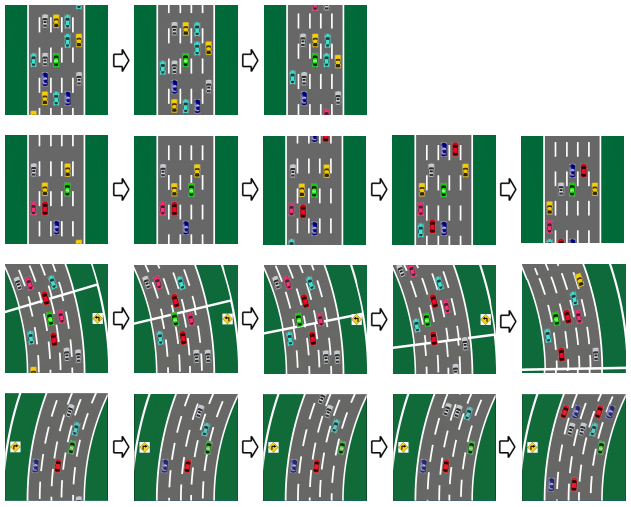

Fig. 6: Overtaking scenarios by implementing π_1^*.

Next, we implemented the policies π_1^* and π_2^* in simulation, and show the overtaking and tailgating behaviors in Figures 6-7, respectively [3].

The first row of Figure 6 shows a scenario where there is a vacant space in front of the HV (green). The HV accelerates to occupy the front space and then maintains its distance behind the yellow vehicle. The second row shows a scenario where there is one vehicle in front of the HV and both the left and right lanes are available for the HV to overtake the front vehicle. Since the road is straight, the HV is free to use either the left or the right lane to complete the overtaking task. One sees from this figure that the HV first switches to the left lane, and then accelerates to overtake the front yellow vehicle, which switches to the right lane, until meeting the blue vehicle in the front. The third scenario shows one vehicle (red) in front of the HV, but the right lane of the HV is occupied by another vehicle (pink). The HV can only use the left lane to overtake the front vehicle. The last scenario shows another driving scenario during cornering, which has one vehicle (cyan) in front of the HV and both the left and right lanes of the HV are available to use for overtaking. The HV first switches to the right lane, which is closer to the inner curb of the corner, and then tries to overtake the cyan vehicle by accelerating. All these driving behaviors in the simulation using π_1^* agree with the desired behaviors.

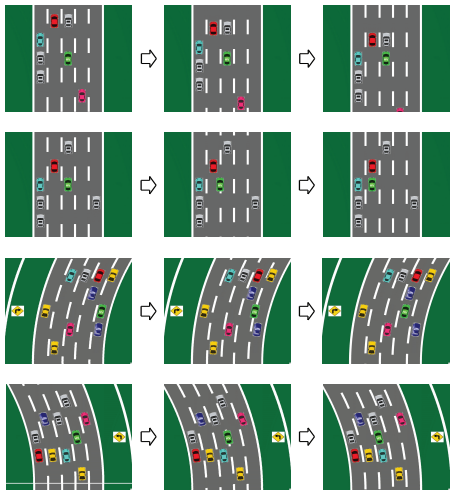

Fig. 7: The tailgating in simulation by implementing $\hat{\pi}_2$.

Figure 7 shows four driving scenarios. The first row shows a driving scenario having a vacant space in front of the HV, and the HV cannot tailgate any EV by changing lanes. The HV accelerates to occupy the space behind the grey vehicle in front of it. The second driving scenario shows that the HV changes the lane to the left to tailgate the red vehicle. The third driving scenario is similar to the second, but it shows a driving behavior during cornering.

[3]Movies for both overtaking and tailgating by implementing $\hat{\pi}_1^*$ and $\hat{\pi}_2^*$ are available at: https://www.youtube.com/watch?v=I3ecd9DXmBQ and https://www.youtube.com/watch?v=lVZcRR-Q2PE

The last driving scenario shows that there are two EVs in front of the HV in the neighboring lanes. By designing w_2 we have constructed a policy π_2^* to tailgate the EV closer to the inner curb in a corner. One sees that the HV changes to the left lane to tailgate the grey vehicle in a left-turn corner. All these driving behaviors agree with the desired tailgating behaviors.

V. Conclusion

We use a stochastic Markov decision process to model the traffic, and achieve desired driving behaviors using reinforcement learning. The definition of the state and the MDP model are flexible and can be used to model traffic with any number of lanes and any number of EVs. We also take the road geometry into consideration such that the driving policy may change depending on the road curvature. By designing the driver's reward function, we are able to show typical driving behaviors such as overtaking and tailgating, using the Q-learning algorithm to learn the corresponding optimal policies. We have demonstrated these policies using a road with five lanes and with each EV implementing a random policy.

Future work will focus on refining the traffic model for vehicles of different sizes and types (e.g., trucks, buses) having, perhaps, different velocities. Another interesting future research topic would be to incorporate explicit or implicit communication between the host vehicle and the surrounding vehicles in traffic.

ACKNOWLEDGMENT

This work is supported by Ford Motor Company and National Science Foundation award CPS-1544814.

References

[1] S. D. Pendleton, H. Andersen, X. Du, X. Shen, M. Meghjani, Y. H. Eng, D. Rus, and M. H. Ang, "Perception, planning, control, and coordination for autonomous vehicles," *Machines*, vol. 5, no. 1, p. 6, 2017.

[2] S. Shalev-Shwartz, N. Ben-Zrihem, A. Cohen, and A. Shashua, "Long-term planning by short-term prediction," *arXiv preprint arXiv:1602.01580*, 2016.

[3] S. Brechtel, T. Gindele, and R. Dillmann, "Probabilistic MDP-behavior planning for cars," in *14th International IEEE Conference on Intelligent Transportation Systems (ITSC)*, Washington, DC, October 5–7 2011, pp. 1537–1542.

[4] Y. Kuwata, J. Teo, G. Fiore, S. Karaman, E. Frazzoli, and J. P. How, "Real-time motion planning with applications to autonomous urban driving," *IEEE Transactions on Control Systems Technology*, vol. 17, no. 5, pp. 1105–1118, 2009.

[5] S. Karaman and E. Frazzoli, "Sampling-based algorithms for optimal motion planning," *The International Journal of Robotics Research*, vol. 30, no. 7, pp. 846–894, 2011.

[6] C. Thorpe, M. H. Hebert, T. Kanade, and S. A. Shafer, "Vision and navigation for the Carnegie-Mellon NAVLAB," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 10, no. 3, pp. 362–373, 1988.

[7] T. Jochem, D. Pomerleau, B. Kumar, and J. Armstrong, "PANS: A portable navigation platform," in *Proceedings of the Intelligent Vehicles' 95 Symposium*, Detroit, MI, September 25-26 1995, pp. 107–112.

[8] B. Paden, M. Čáp, S. Z. Yong, D. Yershov, and E. Frazzoli, "A survey of motion planning and control techniques for self-driving urban vehicles," *IEEE Transactions on Intelligent Vehicles*, vol. 1, no. 1, pp. 33–55, 2016.

[9] J.-w. Choi, R. Curry, and G. Elkaim, "Path planning based on Bézier curve for autonomous ground vehicles," in *World Congress on Engineering and Computer Science*, San Francisco, CA, October 22–24 2008, pp. 158–166.

[10] J.-w. Choi, R. E. Curry, and G. H. Elkaim, "Continuous curvature path generation based on Bézier curves for autonomous vehicles." *International Journal of Applied Mathematics*, vol. 40, no. 2, 2010.

[11] S. Ulbrich and M. Maurer, "Probabilistic online pomdp decision making for lane changes in fully automated driving," in *16th International IEEE Conference on Intelligent Transportation Systems*, Hague, Netherlands, October 6–9 2013, pp. 2063–2067.

[12] C. Katrakazas, M. Quddus, W.-H. Chen, and L. Deka, "Real-time motion planning methods for autonomous on-road driving: State-of-the-art and future research directions," *Transportation Research Part C: Emerging Technologies*, vol. 60, pp. 416–442, 2015.

[13] T. Hastie, R. Tibshirani, and J. Friedman, "Overview of supervised learning," in *The Elements of Statistical Learning*. Springer, 2009, pp. 9–41.

[14] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[15] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. MIT Press Cambridge, 1998, vol. 1, no. 1.

[16] S. Lange, M. Riedmiller, and A. Voigtlander, "Autonomous reinforcement learning on raw visual input data in a real world application," in *International Joint Conference on Neural Networks (IJCNN)*, Brisbane, QLD, Australia, June 10–15 2012, pp. 1–8.

[17] M. Ardelt, P. Waldmann, F. Homm, and N. Kaempchen, "Strategic decision-making process in advanced driver assistance systems," *IFAC Proceedings Volumes*, vol. 43, no. 7, pp. 566–571, 2010.

[18] D. W. Oyler, Y. Yildiz, A. R. Girard, N. I. Li, and I. V. Kolmanovsky, "A game theoretical model of traffic with multiple interacting drivers for use in autonomous vehicle development," in *American Control Conference (ACC), 2016*, Boston, MA, July 6–8 2016, pp. 1705–1710.

[19] N. Li, D. Oyler, M. Zhang, Y. Yildiz, A. Girard, and I. Kolmanovsky, "Hierarchical reasoning game theory based approach for evaluation and testing of autonomous vehicle control systems," in *IEEE 55th Conference on Decision and Control*, Las Vegas, NV, December 12–14 2016, pp. 727–733.

[20] R. Bellman, "A Markovian decision process," *Journal of Mathematics and Mechanics*, pp. 679–684, 1957.

[21] P. Abbeel and A. Y. Ng, "Apprenticeship learning via inverse reinforcement learning," in *Proceedings of the 21st International Conference on Machine Learning*, Banff, Canada, July 4–8 2004, p. 1.

[22] B. D. Ziebart, A. L. Maas, J. A. Bagnell, and A. K. Dey, "Maximum entropy inverse reinforcement learning." in *AAAI*, vol. 8. Chicago, IL, 2008, pp. 1433–1438.

[23] ——, "Human behavior modeling with maximum entropy inverse optimal control." in *AAAI Spring Symposium: Human Behavior Modeling*, 2009, p. 92.

[24] S. Levine, Z. Popovic, and V. Koltun, "Nonlinear inverse reinforcement learning with Gaussian processes," in *Advances in Neural Information Processing Systems*, 2011, pp. 19–27.

[25] S. Levine and V. Koltun, "Continuous inverse optimal control with locally optimal examples," *arXiv preprint arXiv:1206.4617*, 2012.

[26] M. Wulfmeier, P. Ondruska, and I. Posner, "Maximum entropy deep inverse reinforcement learning," *arXiv preprint arXiv:1507.04888*, 2015.

[27] A. Y. Ng and S. J. Russell, "Algorithms for inverse reinforcement learning." in *Proceedings of the 17th International Conference on Machine Learning*, Stanford, CA, June 29 – July2 2000, pp. 663–670.

[28] C. J. C. H. Watkins, "Learning from delayed rewards," Ph.D. dissertation, King's College, Cambridge, 1989.

[29] H. R. Berenji, "Fuzzy Q-learning: a new approach for fuzzy dynamic programming," in *Proceedings of the 3rd IEEE Conference on Fuzzy Systems*, Orlando, FL, June 26–29 1994, pp. 486–491.

[30] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller, "Playing Atari with deep reinforcement learning," *arXiv preprint arXiv:1312.5602*, 2013.

[31] H. Van Hasselt, A. Guez, and D. Silver, "Deep reinforcement learning with double Q-learning." in *AAAI*, 2016, pp. 2094–2100.