Markov probabilistic decision making of self-driving cars in highway with random traffic flow: a simulation study

Yang Guan
Tsinghua University, Beijing, China
Shengbo Eben Li
Department of Automotive Engineering, Tsinghua University, Beijing, China, and
Jingliang Duan, Wenjun Wang and Bo Cheng
Tsinghua University, Beijing, China

Abstract

Purpose – Decision-making is one of the key technologies for self-driving cars. The high dependency of previously existing methods on human driving data or rules makes it difficult to model policies for different driving situations.

Design/methodology/approach – In this research, a probabilistic decision-making method based on the Markov decision process (MDP) is proposed to deduce the optimal maneuver automatically in a two-lane highway scenario without using any human data. The decision-making issues in a traffic environment are formulated as the MDP by defining basic elements including states, actions and basic models. Transition and reward models are defined by using a complete prediction model of the surrounding cars. An optimal policy was deduced using a dynamic programing method and evaluated under a two-dimensional simulation environment.

Findings – Results show that, at the given scenario, the self-driving car maintained safety and efficiency with the proposed policy.

Originality/value – This paper presents a framework used to derive a driving policy for self-driving cars without relying on any human driving data or rules modeled by hand.

Keywords Markov decision process, Decision-making, Dynamic programming, Self-driving cars

Paper type Research paper

1. Introduction

Recently, there has been a steady increase in interest of many researchers toward developing technologies for self-driving cars. This technology has the potential to reshape mobility by enhancing the safety, accessibility, efficiency and convenience of transportation. A major milestone in self-driving cars was the DARPA Urban Challenge. In 2007, six teams finished the event, which demonstrated that fully autonomous urban driving is possible (Buehler *et al.*, 2009). The Google self-driving car and Tesla's Autopilot system have been two popular examples that receive considerable attention since then.

The autonomous driving system can be divided into three layers: environment perception, decision-making and dynamic control. Environment perception detects surroundings in real time via radar, laser light, GPS, odometry and computer vision. The decision-making further understands the environment and predicts the movement of different participants. It, then, conducts maneuver selection and path planning. Finally, dynamic control instructs throttle, brake and steering of self-driving cars. In environment perception, Janai *et al.* (2017) reviewed the current state-of-the-art on several specific aspects in computer vision for autonomous vehicles, including recognition, reconstruction, motion estimation, tracking, scene understanding and end-to-end learning. Patole *et al.* (2017) summarized various aspects of automotive radar signal processing techniques as well as unique problems associated with automotive radars such as pedestrian detection. For the perception of multiple vehicles, Liu *et al.* (2018) presented a novel distributed Bayesian filtering method using measurement dissemination for multiple unmanned ground vehicles with dynamically changing interaction topologies. In decision-making, an existing method of motion prediction and risk assessment for intelligent vehicles based on the semantics used to define motion and risk was reviewed by Lefèvre *et al.* (2014).

The current issue and full text archive of this journal is available on Emerald Insight at: **www.emeraldinsight.com/2399-9802.htm**

Journal of Intelligent and Connected Vehicles
1/2 (2018) 77–84
Emerald Publishing Limited [ISSN 2399-9802]
[DOI 10.1108/JICV-01-2018-0003]

Received 31 January 2018
Revised 2 April 2018
3 May 2018
3 June 2018
6 July 2018
Accepted 13 August 2018

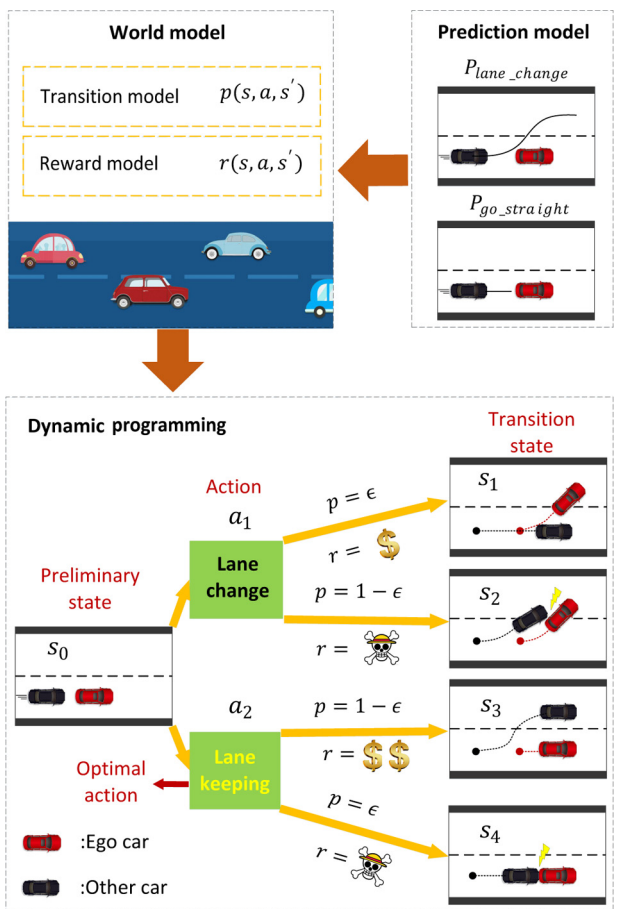
Paden *et al.* (2016) proposed a structured decision-making in the contemporary autonomous driving system into route planning, behavioral decision-making, local motion planning and feedback control. He also summarized the state-of-the-art currently available methods on planning and control algorithms for urban environments. In dynamic control, Carvalho *et al.* (2015) presented an overview of control design methods that systematically handle uncertain forecasts for autonomous and semi-autonomous vehicles. For the control of multiple vehicles, this paper (Li *et al.*, 2017c) introduces a decomposition framework to model, analyze and design the platoon system from the perspective of multiagent consensus control. For the control of multiple vehicles, Zheng *et al.* (2017) presented a distributed model predictive control algorithm for longitudinal automation of connected vehicles with unidirectional topologies. Li *et al.* (2017b) presented a robust distributed control method for multiple vehicles with bounded parameter uncertainty and a broad spectrum of interaction topologies.

This paper focuses on the decision-making, which is a tricky problem to tackle owing to the highly dynamic, stochastic and uncertain nature of the traffic environment. The finite state machine (FSM) has become the most common approach for decision-making of self-driving cars since the Urban Challenge in 2007. It models finite situations of the traffic environment manually and obtains policies based on rules. These situations and rules make the decision system explicit, but it needs an experienced expert to design them to obtain good performance. The champion of 2008 Urban Challenge Boss (Urmson *et al.*, 2008) used this approach in its behavioral layer. It defined three high-level behaviors in advance, including lane driving, intersection handing and goal selection. Each of them corresponds to several low-level behaviors. The decision system chose these behaviors by rules prescribed in advance when the car ran at different situations. The runner-up Junior (Montemerlo *et al.*, 2008) used FSM for switching among 13 driving situations and invoking exception behaviors to overcome stuckness. Besides, Odin (Bacha *et al.*, 2008) ran with appropriate behaviors under the current situation using a system of the hierarchical FSM. The system is capable to distinguish among intersection, parking lot and normal road scenarios. Similarly, Furda *et al.* (Furda and Vlacic, 2011) presented a multiple-criteria decision-making approach to divide the decision-making task into two stages: the first one determined feasible maneuvers based on Petri nets, whereas the second used a multivariate utility function to select the most appropriate one. The FSM is simple and effective in given situations. However, it does not explicitly consider environment uncertainties and thus cannot be applied in a dynamic traffic scenario. Besides, it requires classifying situations and modeling policies by hand, which fails to make a decision under unusual situations.

Compared with explicit decomposition of the problem, Chen *et al.* (2015) trained a convolutional neural network (CNN) model on 484,815 images collected and labeled when playing a car racing video game TORCS for 12 h. The model mapped an input image to a small number of key perception indicators that is then sent in the designed controller. Bojarski *et al.* (2016) trained a CNN model, mapping raw pixels from a single front-facing camera directly to steering commands. The training data were collected by driving on a wide variety of roads and in a diverse set of lighting and weather conditions. About 72 h of driving data were collected and sampled at ten frames per second, which is used for training after being augmented further. The system can learn how to drive on local roads or highways with lane marking or not. The paper (Bojarski *et al.*, 2017) explained how the neural-network-based system called PilotNet learns and makes decisions. They developed a method for determining which elements in the road image influence the steering decision the most. The paper also showed that PilotNet can learn more subtle features that are hard to program and anticipate by engineers. These end-to-end approaches simultaneously optimize all processing steps instead of human-selected intermediate criteria. It will performance better, especially for smaller systems, but requires huge amounts of data to get good performance.

This paper proposes a probabilistic decision-making method, which can be applied in a dynamic traffic environment while considering safety, efficiency and comfort simultaneously. The driving task was first formulated as the Markov decision process (MDP) by defining the environment state space, agent action space. Then, it built a state transition model and a reward model by using a prediction model of surrounding cars. The optimal policy was then automatically deduced using the value iteration method of dynamic programming (DP). The simulation results show the preset goal can be achieved. The framework of the proposed method is shown in Figure 1. This paper provides a probability decision-making approach that is neither dependent on human driving data nor limited to only a variety of rules

Figure 1 Framework of the method



modeled by hand. In addition, the formulation of the driving task under a two-lane highway scenario is presented, including state discretization and transition model estimation.

The remainder of the paper is structured as follows: In Section 2, a basic knowledge of the MDP is presented. Section 3 presents a MDP formulation for the driving task. Section 4 analyzes the result policy and provides explanations and intuitions about the MDP method. Finally, Section 5 concludes with remarks on the main work of this paper and potential areas for future research.

2. Basic knowledge of MDP

This section will give an overview about the used terminology. There are five basic elements of MDP, that is the tuple $(\mathcal{S}, \mathcal{A}, r(s,a,s'), p(s,a,s'), \gamma)$, whereas $\mathcal{S}$ denotes the set of all states, $\mathcal{A}$ denotes the set of all actions, $r(s,a,s') \in \mathcal{R}$ denotes the expected immediate reward on transition from state s to s' under action a, where $s, s' \in \mathcal{S}$, $a \in \mathcal{A}$, and $\mathcal{R}$ denotes set of all possible rewards. And $p(s, a, s')$ denotes the probability of transition to state s' from state s taking action a.

A policy π is a stochastic rule by which the agent selects actions as a function of states. The agent's objective is to maximize the amount of reward it receives over time. That is, finding an optimal policy $\pi^\star$ satisfying formula (1), whereas $v_\pi(s)$ denotes the expected return from state s using policy π.

$$\pi_* = \arg\max_\pi v_\pi(s) \tag{1}$$

For all $s \in \mathcal{S}$.

The Bellman optimality equation (2) is a special consistency condition that the optimal value functions must satisfy and that can, in principle, be solved for the optimal value functions, from which an optimal policy can be determined with the value iteration method.

$$\pi_* = \arg\max_a \sum_{s',r} p(s', r|s, a)\left[r + \gamma v_{\pi_*}(s')\right] \tag{2}$$

3. MDP formulation for driving task

MDP is a mathematically formulation for decision and control problems with uncertain system behavior. To derive the optimal policy using it, a tuple including state, action, transition model, reward model and discounting is first required.

3.1 Environment state space

Variables selected in states are supposed to contain the complete environmental information, such as properties of road, own car and other cars, which is required by the agent decision. However, with the number of variables included in a state rising, the number of states increases exponentially, which causes computation and memory problems. As a result, it is necessary to do simplification work to minimize the number of these variables. Consider there is only one car of interest and assume that cars are always parallel to the lane line. In addition, distance is more worthy of concern than absolute position in the longitudinal direction. Consequently, we formulate a tuple as a state.

$$(\Delta x_{lon}, x_{lat\_ego}, x_{lat\_veh}, v_{lon\_ego}, v_{lon\_veh}) \in \mathcal{S}_c \tag{3}$$

where Δx_{lon} represents the longitudinal distance between the ego car and the other car, and it is positive when the ego car is behind the other car. x_{lat_ego}, x_{lat_veh}, denote lateral positions of the ego car and the other car, and v_{lon_ego}, v_{lon_veh} denote the longitudinal velocity, respectively.

MDP needs a discrete description. For variables related to distance, we discrete it by dividing the road into no overlapping areas with equal distance along longitudinal and lateral directions. Besides, the speed is discreted with a fixed interval. The state space is discretized as shown in Figure 2.

3.2 Agent action space

Action space has all decisions that we can make. There are two main layers of decision-making architecture for intelligent cars, that is driving behavior layer and trajectory planning layer. Therefore, decisions can be deduced from all trajectories planned for all driving behaviors. As a result, the action space should contain all these trajectories. First, we define our behavior set. Li *et al.* (2017a) categorized driving behavior in highway traffic into 12 maneuver states. Here we define behavior set $\mathcal{B}$ as:

$$\mathcal{B} = \mathcal{W} \times \mathcal{D} \tag{4}$$

where $\mathcal{W}$ denotes the basic behavior set and $\mathcal{D}$ denotes the degree set of basic behavior radicalism. They are defined as:

$$\mathcal{W} = \{go\ straight, turn\ left, turn\ right\} \tag{5}$$

$$\mathcal{D} = \{very\ radical, medium\ radical, normal, medium\ cautious, very\ cautious\} \tag{6}$$

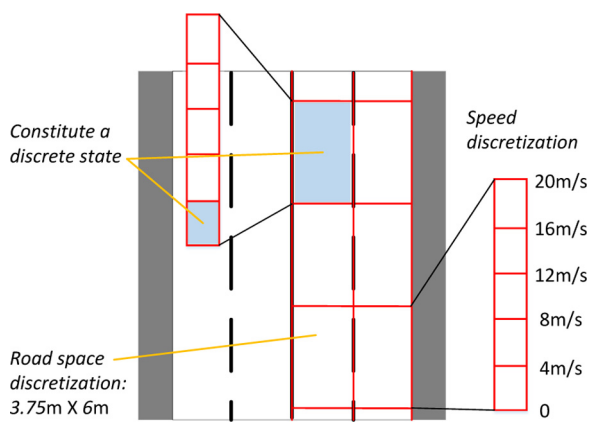
Then, we use a tuple to represent a trajectory in the action space,

$$(a_{lon}, v_{lat}, t) \in \mathcal{A}_c \tag{7}$$

where a_{lon} denotes the longitudinal acceleration, v_{lat} denotes the lateral speed and t denotes the time that actions go through.

Every $b \in \mathcal{B}$ corresponds a subset of $\mathcal{A}_c$. We can discretize all these subsets the same way as state discretization. a_{lon} and v_{lat}

Figure 2 State discretization



can be discretized within their scope of experience. However, a reasonable range should be determined for the discretization of t. It cannot be too long or too short, so 1-3 s is appropriate.

3.3 Environment transition model

3.3.1 Prediction model

Transition model $p(s, a, s')$ describes the probability of transition from state s to s' under action a. Thus, a prediction model for the other car is needed to obtain the transferred state. Suppose the scenario has two lanes and the driving behavior set includes lane keeping and lane change. First, the probability of driving behavior is predicted using statistic data, and then, the motion prediction is carried out according to the corresponding trajectory of that driving behavior.

3.3.2 Transition model

Continuous state space $\mathcal{S}_c \subseteq R^n$ and discrete state space $\mathcal{S}$ can be linked by a random variable $I_s : \mathcal{S}_c \to \mathcal{S}$, which is defined on a probability space $(\mathcal{S}_c, \mathcal{F}, P)$. $\mathcal{F}$ is the Borel σ-algebra on $\mathcal{S}_c$. (Royden and Fitzpatrick, 1968) Thus, for every discrete state $s \in \mathcal{S}$, its corresponding continuous state set is $I_{\mathcal{S}}^{-1}(s) = \{s_c : I_s(s_c) \in s\}$. Then, we can acquire the probability measure in probability space $\left(S, \mathcal{F}', P^{I_s}\right)$, where $\mathcal{F}'$ is the power set of S. For an event $D \in \mathcal{F}'$, its probability of occurrence can be calculated by formula (8).

$$P^{I_s}(D) = P\left(I_{\mathcal{S}}^{-1}(D)\right) = \int_{s_c \in I_{\mathcal{S}}^{-1}(D)} p(s_c) \tag{8}$$

In the same way, action sets can also be associated by random variable $I_a : \mathcal{A}_c \to \mathcal{A}$. Therefore, the probability of transition from state $s \in \mathcal{S}$ to $s' \in \mathcal{S}$ under action $a \in \mathcal{A}$ can be solved by formula (9).

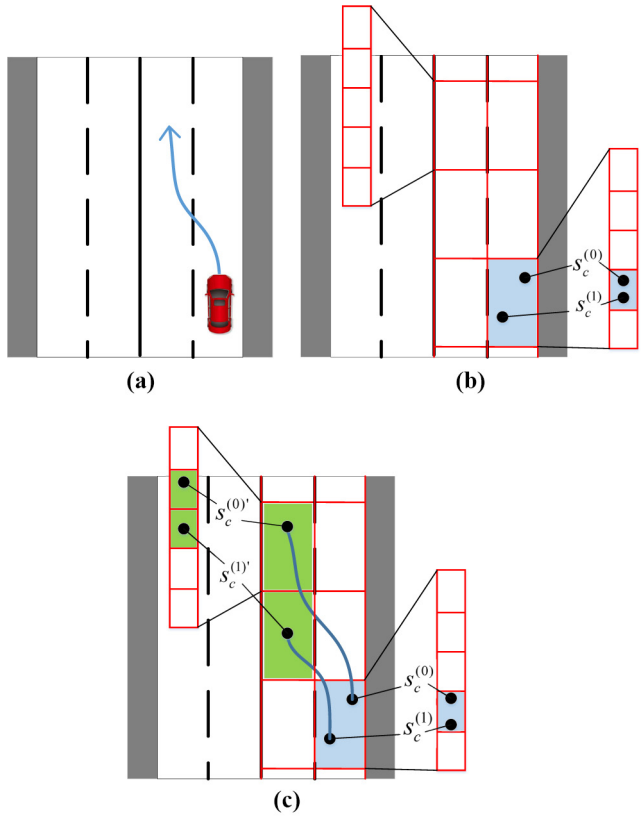
$$\begin{aligned} p(s'|s,a) &= P\left(I_{\mathcal{S}}^{-1}(s') | I_{\mathcal{S}}^{-1}(s'), I_{\mathcal{A}}^{-1}(a)\right) \\ &= \int_{s_c' \in I_{\mathcal{S}}^{-1}(s')} \int_{s_c \in I_{\mathcal{S}}^{-1}(s)} \int_{a_c \in I_{\mathcal{A}}^{-1}(a)} p\left(s_c' | s_c, a_c\right) \end{aligned} \tag{9}$$

It is hard to solve the transition probability using formula (9) directly. However, it is feasible to approximate it with the Monte Carlo method. We first sample Q state points with weight on the continuous state space $I_{\mathcal{S}}^{-1}(s)$ corresponding to the discrete state s. Using action a and our prediction model, we get the Dirac distribution of s_c' for every sample point. Then, the distribution of s_c' given s and a is obtained by weighted average on all the samples. It is also a Dirac distribution. Finally, we can get $p(s'|s,a)$ by summing the probability on all $s_c' \in I_{\mathcal{S}}^{-1}(s')$. Figure 3 shows a sampling instance of turning left.

3.4 Environment reward model

The reward model $r(s, a, s')$ can be obtained just the same way as $p(s, a, s')$. But how many rewards the agent is able to get after adopting action a need to be determined first. This reward has a relationship with the driving goals, which usually include safety, efficiency, comfort, economy and compliance with traffic rules and daily driving habits. We set the reward a large negative number when the ego car goes into a bad terminal state to ensure safety. Bad states include colliding with the other car, driving out of the road and being dumped by the other car. For efficiency, a large positive number is assigned to the reward when the ego car dumps the other car, which is the good terminal state. And we achieve other goals by designing rewards as formula (10) when the ego car does not transfer to a terminal state.

Figure 3 State space sampling and transition



Notes: (a) Scenario; (b) sampling; (c) transition

$$r = \sum_{t=1}^{t/\Delta t} \Delta t (f_{com} a_{lon} + f_{ove} v_{lat} + f_{\text{obe}}(!Is_{right})) \tag{10}$$

To achieve the purpose of comfort and economy, a negative factor f_{com} is added in front of the acceleration to punish the large value of it. To encourage overtaking moves, a positive factor f_{ove} is added in front of the lateral velocity. Because we need the whole action execution process to infer the environment model, an action is decomposed into several steps to implement while a step time is Δt. To keep the ego car in the right lane as far as possible, we give it a negative value f_{obe} when it is in the left lane in a step but nothing when it is not. By multiplying all the three terms mentioned above by the step time and summing over them on all the steps during the action implementation, the reward can be obtained.

4. Policy evaluation

For the evaluations, we use a two-lane highway scenario. The self-driving agent has to cope with the other vehicle and

manage tasks like overtaking and avoiding collision. A detailed analysis of the results and tests in a two-dimensional simulation environment has been done.

4.1 Analysis of the results

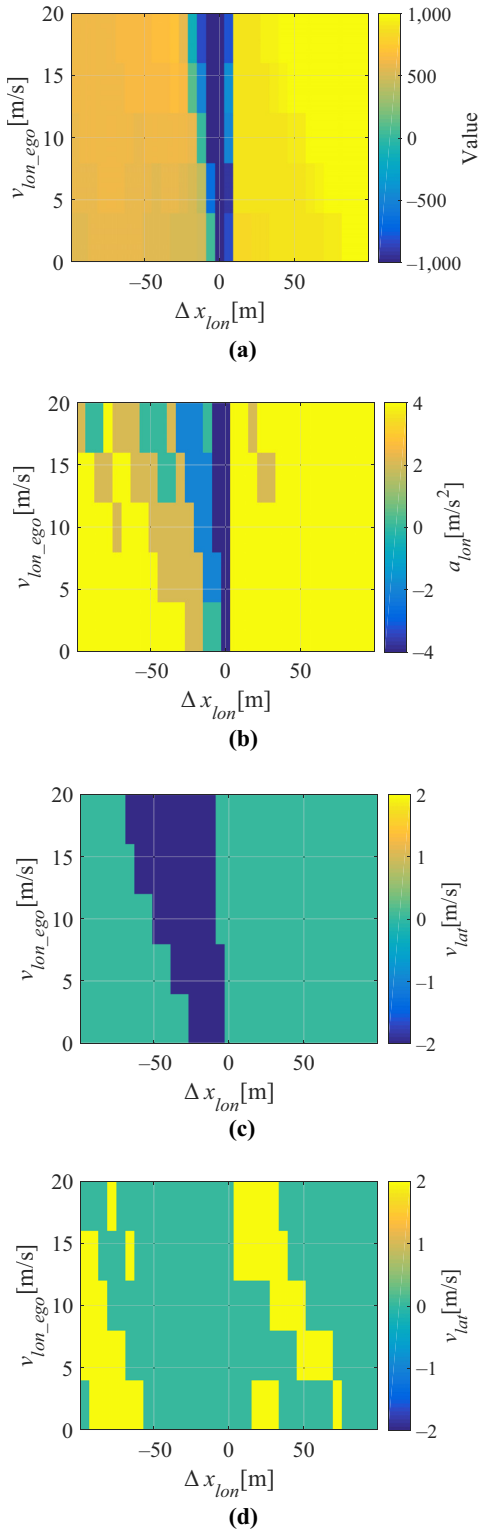
After building the MDP model, we applied the value iteration method to get the optimal policy. When the ego car transfers to a bad terminal state, it is given a reward of −1000. On the contrary, a reward of 1000 is given when it goes into a good terminal state. Otherwise, we assign the reward using formula (10) while setting $f_{com} = 0$, $f_{ove} = 0$, $f_{obe} = -30$. Besides, we set discounting factor γ to 0.9. There are 64 samples of the ego car and the other car. Hence, the number of samples is 64×64. The results are as shown in Figure 4 and Figure 5. Figure 4 shows the optimal policy when the other car is in the right lane and its speed is between 0 and 4 m/s. Figure 4a, 4b and 4c, respectively, represent the state value, the optimal acceleration and the optimal lateral speed when the ego car is also in the right lane, whereas Figure 4d represents the optimal lateral speed when the ego car is in the left lane. Each sub-figure corresponds to values under 165 states, which consists of 33 discrete values of Δx_{lon} horizontally and five discrete values of x_{lat_veh} vertically.

As can be seen in Figure 4a, when $\Delta x_{lon} > 10$ m, the state value increases with the vehicle speed increasing. This is because when the ego car is beyond the other car, it can reach the good terminal state to get the 1000 reward. Because future rewards decay exponentially with respect to steps taken from the current state, the ego car arrives faster, the greater return it gets, and that is why states with a greater speed have a higher value. By contrast, when $\Delta x_{lon} > -30$ m and <0, the state value decreases with the increase of the vehicle speed. This is because when the ego car is close to its front car, the greater its speed, the more likely it collides and gets a reward of −1000, the lower the state value.

The optimal policy is the policy that leads to highest values for all states. The Bellman equation shows that the value of a state consists of instantaneous reward and the value of the following state. Therefore, policy selection is a trade-off among three factors including efficiency, safety and avoiding staying in the left lane too long. As shown in Figure 4b, when the ego car is behind and its speed is small, efficiency is more worthy of attention than safety. Thus, a large acceleration is adopted. However, with the speed increasing, the agent is more concerned about preventing collision than being faster. As a result, a small acceleration or even a negative one is taken in this case. On the other hand, when $\Delta x_{lon} < 0$, optimal accelerations decrease with $|\Delta x_{lon}|$ decreasing also owing to safety considerations. When the ego car goes to the front, there is almost no collision avoidance problem. In addition, there is no comfort or economy consideration in the reward function. The goal is to reach the destination as soon as possible, so the maximum acceleration is chosen, namely, 4 m/s^2.

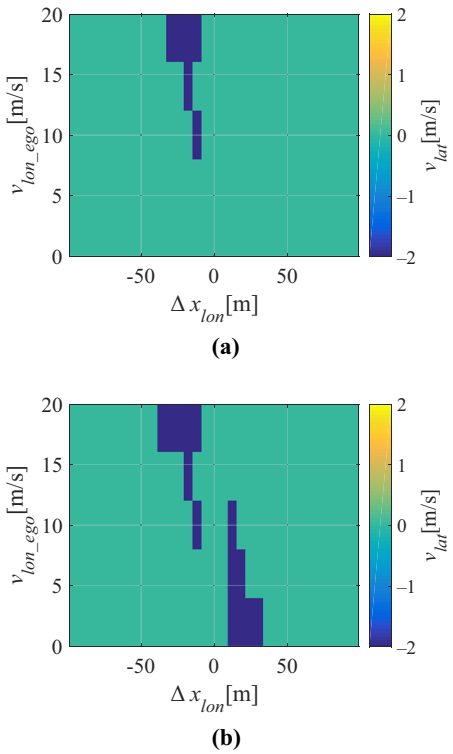
For the lateral speed selection, it is still the trade-off among the three factors. As can be seen in Figure 4c, when the ego car runs behind, lateral speed changes from 0 to 2 m/s with the increasing of the vehicle speed. That is because the larger the speed, the more unsafe the agent feels, so it switches to the left lane to avoid collision. For the same reason, the optimal lateral speed changes from 0 to 2 m/s with the distance decreasing.

Figure 4 Optimal policy when the other car is at the right lane with speed 0-4 m/s



Notes: (a) Maximal value; (b) acceleration; (c) lateral speed (ego on the right lane); (d) lateral speed (ego on the left lane)

Figure 5 Comparison of the different prediction model



Notes: (a) Complete prediction model; (b) only lane keeping prediction

However, it changes back to 0 when the distance continues to decrease. This is because in emergency, owing to physical constraints of the tire, the lateral force is turned to 0 to make the longitudinal force the maximal value. When the ego car runs at the front, there are no collision worries, so the lateral speed always equals 0.

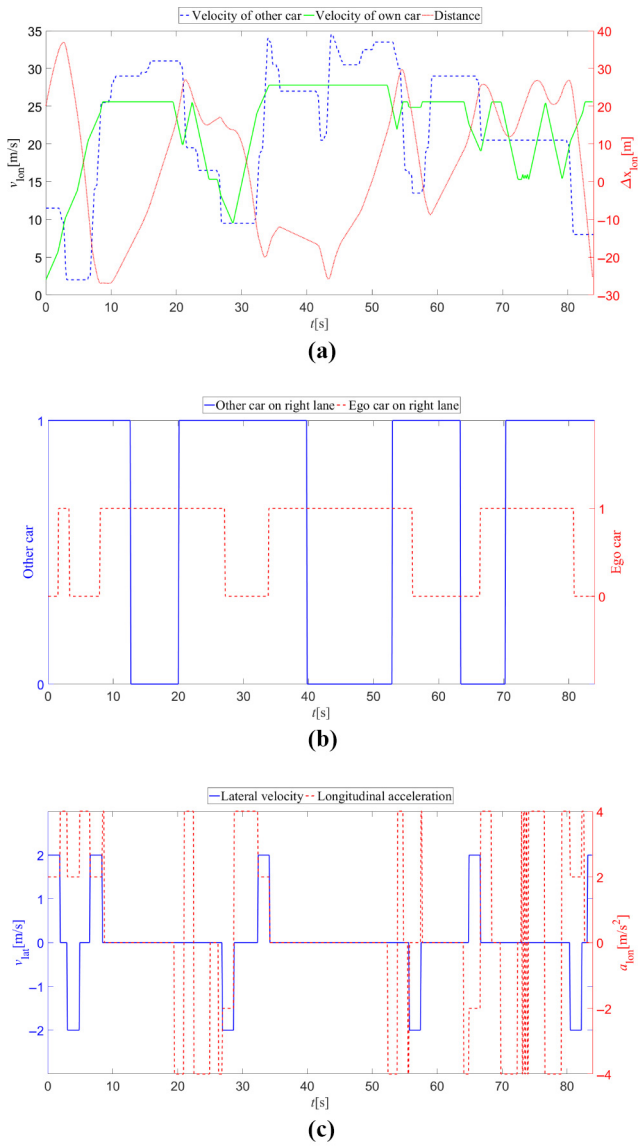
In Figure 4d, we can see when the ego car runs behind, the greater the distance and the lower the speed, the more the tendency of changing lanes. In these cases, it is the main consideration to avoid staying in the overtaking lane. Otherwise, we care more about the efficiency and safety. Then, when it runs at the front, with the distance increasing, the focus of our attention changes from safety to rule compliance, and then efficiency. And that is why the optimal lateral speed changes from 0 to 2 m/s, and then 0.

Figure 5a shows the importance of a more sophisticated prediction model. The situation is basically the same as the above example, except the other car drives on the right lane with a speed between 12 and 16 m/s. Obviously, the safety distance shrinks compared with the case where the other car is in low speed, because there is more time to brake. The difference between 5a and 5b is 5a has a complete driving behavior set in its prediction model, whereas 5b only has lane keeping behavior. The missing uncertainty leads to an extra lane changing behavior when the rear car is approaching, because the ego car predicts the rear car will go straight so that it does this to avoid collision. However, this is extremely dangerous when it is really driving because the rear car is very likely to overtake in that situation. The complete prediction model is able to forecast this, and consequently, the ego car in Figure 5a holds lane.

4.2 Simulation test

The decisions of the agent with the other car were practically evaluated in a two-dimensional simulation environment. The speed limit is 30 m/s for the ego car. Several tests were done when the other car was being controlled manually with acceleration and the steering wheel. One of the simulation processes is shown in Figure 6. In this process, the other car is being controlled to behave at random, as the blue lines shown in Figure 6a and 6b. We can see the agent showed reasonable

Figure 6 Simulation results



Notes: (a) Distance and velocity; (b) whether the other car or the ego car on the right lane; (c) maneuver selection of the ego car

behavior. Sometimes the controlled car fast approached the ego car in the right lane and overtook it, just as what happened around 10 s. In this situation, the ego car would keep a constant speed and stay in the right lane rather than turning left to avoid collision. Sometimes the ahead controlled car suddenly decelerated, as what happened between 20 s and 30 s. Under this circumstance, it can be seen that the ego car chose to decelerate too and when it slowed down, it chose to turn left and overtook the front car. Besides, when the high-speed controlled car overtook the ego car from the left lane and turned to the right lane suddenly, the ego car would decelerate right away, just as shown around 20 s, 50 s and 62 s.

It can be seen that the results obtained here are consistent with previous analyses. The deduced policy can adapt to any movement of the surrounding car and make the ego car drive safely and efficiently.

4.3 Comparison to existing methods

From the above results, it can be seen that the MDP defines per step reward and uses the DP method to choose behavior in each state to get the maximal expected total reward in the future. To implement the DP, it first needs to estimate the transition model through the Monte Carlo method. Compared with the FSM method, which chooses behavior by rules, the MDP does not need too much experience to design the whole decision system but a little knowledge to design the per step reward, which is related to the driving goals. Compared with the CNN method, which chooses action by a CNN trained on a large number of labeled driving images, the whole process of the MDP does not need any driving data but an environment model used in equation (2) to get the policy by iteration. As a result, the performance of the MDP is restricted by the precision of the estimated model. Besides, a large state space will lead to expensive computing cost when using the DP. In conclusion, the MDP is more suitable for problems where the precise environment model is available and the surrounding traffic environment is not too complex.

5. Conclusion

This paper presents a method that can automatically deduce the optimal behavior for autonomous driving. This method formulates driving tasks as the MDP and integrates a sophisticated motion prediction model of the surrounding car, in which predictions in continuous space and MDP planning in discrete space are combined by means of the probability method. The deduced optimal policy in the given two-lane highway scenario is evaluated analytically and used in a two-dimensional simulation environment. The results show that it behaves reasonably to achieve safe and efficient driving, such as braking when the front car slows down, overtaking the front car when it drives slow and keeping its lane rather than changing lanes when the rear car is fast approaching. This method could be applied in standard scenarios, such as highway and park driving, so that there is no need to model policies by hand comparing with rule-based methods.

But there are several issues that need to be improved. First of all, the time needed rises sharply as the dimension of the state space increases owing to high computational complexity. Because lots of states are virtually impossible to meet in practice, it is reasonable to compute on-line when a new state is encountered rather than doing all the computation off-line. In addition, the result shows that fixed discretization is too fine for states whose values are close and discrete interval is too large for states whose values have a significant difference. A more efficient discretization method is required. Finally, to use the decision-making model in reality, a more detailed vehicle dynamic model and a traffic model need to be included; besides, the approximation function should be used to represent the value function in the continuous state space to enhance generalization of the policy and reduce the computation cost in large-scale decision problems.

References

Bacha, A., Bauman, C., Faruque, R., Fleming, M., Terwelp, C., Reinholtz, C., Hong, D., Wicks, A., Alberi, T., Anderson, D. and Cacciola, S. (2008), "Odin: team VictorTango's entry in the DARPA Urban Challenge", *Journal of Field Robotics*, Vol. 25 No. 8, pp. 467-492.

Bojarski, M., Del Testa, D., Dworakowski, D., Firner, B., Flepp, B., Goyal, P., Jackel, L.D., Monfort, M., Muller, U., Zhang, J. and Zhang, X. (2016), "End to end learning for self-driving cars", *arXiv Preprint arXiv*, Vol. 1604, p. 07316.

Bojarski, M., Yeres, P., Choromanska, A., Choromanski, K., Firner, B., Jackel, L. and Muller, U. (2017), "Explaining how a deep neural network trained with end-to-end learning steers a car", *arXiv Preprint arXiv*, Vol. 1704, p. 7911.

Buehler, M., Iagnemma, K., and Singh, S. (2009), *The DARPA Urban Challenge: Autonomous Vehicles in City Traffic*, Springer, Berlin, Vol. 56.

Carvalho, A., Lefévre, S., Schildbach, G., Kong, J. and Borrelli, F. (2015), "Automated driving: the role of forecasts and uncertainty a control perspective", *European Journal of Control*, Vol. 24, pp. 14-32.

Chen, C., Seff, A., Kornhauser, A., and Xiao, J. (2015), Deep driving: learning affordance for direct perception in autonomous driving. in *Proceedings of the IEEE International Conference on Computer Vision*, pp. 2722-2730.

Furda, A. and Vlacic, L. (2011), "Enabling safe autonomous driving in real-world city traffic using multiple criteria decision making", *IEEE Intelligent Transportation Systems Magazine*, Vol. 3 No. 1, pp. 4-17.

Janai, J., Güney, F., Behl, A. and Geiger, A. (2017), "Computer vision for autonomous vehicles: problems, datasets and state-of-the-art", *arXiv Preprint arXiv*, Vol. 1704, p. 05519.

Lefèvre, S., Vasquez, D. and Laugier, C. (2014), "A survey on motion prediction and risk assessment for intelligent vehicles", *Robomech Journal*, Vol. 1 No. 1, p. 1.

Li, G., Li, S.E., Cheng, B., and Green, P. (2017a), "Estimation of driving style in naturalistic highway traffic using maneuver transition probabilities", *Transportation Research Part C*, Emerging Technologies, Vol. 74, pp. 113-125.

Li, S.E., Qin, X., Li, K., Wang, J. and Xie, B. (2017b), "Robustness analysis and controller synthesis of homogeneous vehicular platoons with bounded parameter

uncertainty", *IEEEASME Transactions on Mechatronics*, Vol. 22 No. 2, pp. 1014-1025.

Li, S.E., Zheng, Y., Li, K., Wu, Y., Hedrick, J.K., Gao, F. and Zhang, H. (2017c), "Dynamical modeling and distributed control of connected and automated vehicles: challenges and opportunities", *IEEE Intelligent Transportation Systems Magazine*, Vol. 9 No. 3, pp. 46-58.

Liu, C., Li, S.E., Yang, D. and Hedrick, J.K. (2018), "Distributed Bayesian filter using measurement dissemination for multiple unmanned ground vehicles with dynamically changing interaction topologies", *Journal of Dynamic Systems, Measurement, and Control*, Vol. 140 No. 3, p. 030903.

Montemerlo, M., Becker, J., Bhat, S., Dahlkamp, H., Dolgov, D., Ettinger, S., Haehnel, D., Hilden, T., Hoffmann, G., Huhnke, B., *et al.* (2008), "Junior: the Stanford entry in the urban challenge", *Journal of Field Robotics*, Vol. 25 No. 9, pp. 569-597.

Paden, B., Čáp, M., Yong, S.Z., Yershov, D. and Frazzoli, E. (2016), "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.

Patole, S.M., Torlak, M., Wang, D. and Ali, M. (2017), "Automotive radars: a review of signal processing techniques", *IEEE Signal Processing Magazine*, Vol. 34 No. 2, pp. 22-35.

Royden, H.L. and Fitzpatrick, P. (1968), *Real Analysis*, Vol. 2, Macmillan, New York, NY, Vol. 2.

Urmson, C., Anhalt, J., Bagnell, D., Baker, C., Bittner, R., Clark, M., Dolan, J., Duggins, D., Galatali, T., Geyer, C. and Gittleman, M. (2008), "Autonomous driving in urban environments: boss and the urban challenge", *Journal of Field Robotics*, Vol. 25 No. 8, pp. 425-466.

Zheng, Y., Li, S.E., Li, K., Borrelli, F. and Hedrick, J.K. (2017), "Distributed model predictive control for heterogeneous vehicle platoons under unidirectional topologies", *IEEE Transactions on Control Systems Technology*, Vol. 25 No. 3, pp. 899-910.

Corresponding author

Shengbo Eben Li can be contacted at: lishbo@tsinghua.edu.cn