Merge Maneuver by Autonomous Vehicle using Reinforcement Learning in Dense Traffic

Prof. Dr. Sunil R. Dhore
Department of Computer Engineering
Army Institute of Technology

Abhinay
Army Institute of Technology, Pune

Akash Kumar Singh
Army Institute of Technology, Pune

Amit Kumar Yadav
Army Institute of Technology, Pune

Garima
Army Institute of Technology, Pune

***Abstract*—Autonomous vehicles are good at mundane driving tasks. Merging in traffic is challenging task. Humans look for a empty slot in traffic and guess the behaviour of other drivers on road to perform the merge maneuver and avoid deadlock. This dynamic makes it even more challenging for autonomous vehicles. An autonomous vechicle cannot consider other cars on the road as moving objects; the human behaviour has also to be considered.**

Keywords : Autonomous vehicle, Partially Observable Makarov Decision Process, Reinforcement learning, belief, driver model.

I. Introduction

Autonomous or self-driving vehicles is rapidly developing field mainly due to developments in machine learning techniques and advantages it provides that includes improved safety, reduced congestion, lower emissions and greater mobility. A lot of research has happened in motion planning and obstacle avoidance algorithms using probabilistic methods and it remains an active area of research with developments in machine learning. While transporting passengers or goods from a given origin to agiven destination, motion planning methods incorporate searching for apath to follow, avoiding obstacles and generating the best trajectory that ensures safety, comfort and efficiency.

The goal for motion planning for autonomous vehicle is to select a collision free trajectory that fulfills the mission goal, reaching the destination as fast as possible, while at the same time taking into account the effect of our own motion on surrounding traffic participants. In congested traffic, it is not always for a vehicle to progress along its route without any negative effect on other participants such as requiring them to slow down slightly.

The remainder of the paper starts with a presentation of related work (Section II). It is followed by a presentation of approach (Section III). Based on that, we present implementation (Section IV) and experiment (**??**). Finally, a conclusion is drawn (Section VI).

II. Related Work

In dense traffic scenarios vehicle may come to a halt. With no movement al all, it freezes. Cooperation and collision avoidance models are required to prevent deadlock scenarios aka freezing robot problem [2]. Planning for autonomous

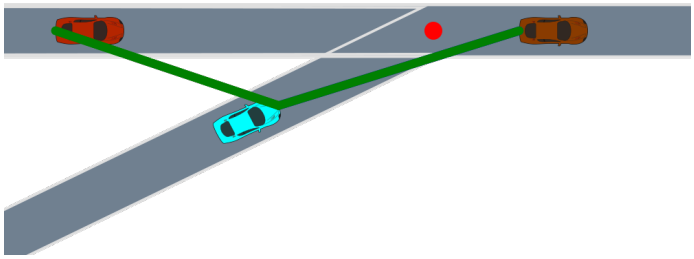

Figure 1. Merging Scenario

systems is challenging because of interactions with other dynamic systems in environment such as humans. Probabilistic interaction models for online planning address this problem [3] - [6]. Online planners take decisions in real time as they perceive the environment. And so require heavy computation when taffic is dense. Online planners take decisions or plan based on inputs from environment that it simulates upto some future time. This future time horizon is shortened due to computation complexity [4], [7]. The behaviour policy of agent depends on the environment model [8]. Policy performance increases when the planning algorithm has access to information about the driver internal state in lane changing scenarios [8].

Mutual influence between human and agent has been studied using using data-driven approaches, probabilistic models, inverse reinforcement learning, rule-based methods, or game theoretic frameworks [3], [5], [6], [9], [10]. Schmerling et al. demonstrated a datadriven approach to learn the interaction model on a traffic weaving scenario involving two agents [5] but is not suitable for dense traffic scenarios where more than two traffic participants are interacting. Lane changing [11], and intersection navigation [12], [13] are two of many driving scenarios where reinforcement learning promises good results.

Here, we test a reinforcement learning agent's ability to interact with environment to successfully perform a merge maneuver in dense scenarios. There is uncertainity associated with behaviour of human driver. That leaves the autonomous system devoid of an important input that can help it perform its task efficiently. Deep Reinforcement learning can capture this

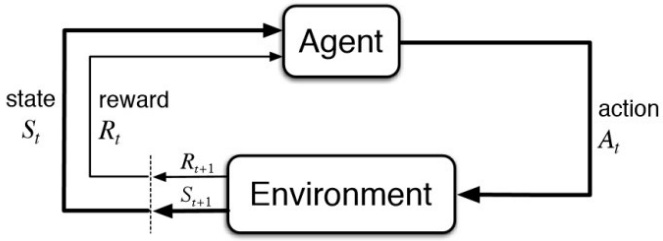

Figure 2. Reinforcement Learning

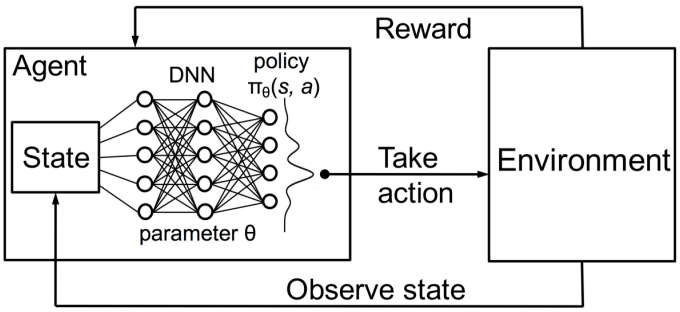

Figure 3. Deep Reinforcement Learning

uncertainity. We address this uncertainity using cooperation level of driver and maintain a belief over it. Simulation results shows that an RL agent using belief states as input yields better performance than standard RL techniques as well as an online planning solver. In addition, we propose a simple rule-based behavior parameterized by a cooperation level to model the reaction of vehicles on the main lane to the merging vehicle.The environment is modeled such that merging car has lower priority than vehicle on main lane. We consider a dense traffic scenario where gap between cars is less(below 2 m) and speed is slow(around 5 m/s).

III. Proposed Approach

Merging scenario can be modeled as partially observable markov decision process(POMDP) states, observations, actions, reward and transition.

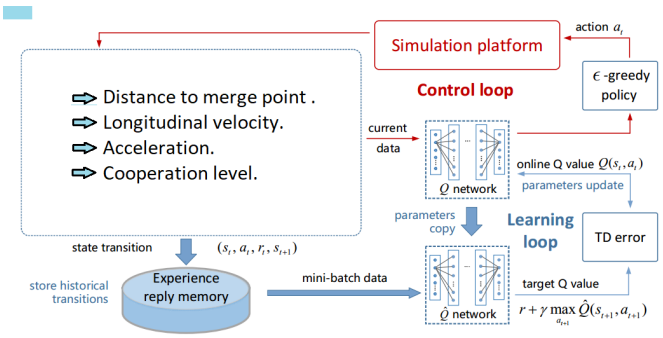

Figure 4. Learning Loop

A. Merging Scenario POMDP

1) States : Vehicle state is combination of its physical state and internal state. The physical state of a vehicle corresponds to its distance to the merge point, its longitudinal velocity, and its acceleration. In this work, the behavior is characterized by a single parameter c, the cooperation level. We let $s_t^i = (x_t^i, v_t^i, a_t^i, c_t^i)$ be the state of vehicle i at time t. The state of the ego vehicle, s_t^ϵ , does not contain the behavior parameter. The complete state of the environment consists of the collection of the individual states of each vehicle present: $s_t = (s_t^\epsilon, s_t^1, ..., s_t^{n_t})$ where n_t is the number of vehicles present at time t.
2) Observations : Sensor noise is not considered. There is a limit to what can be observered and how far. Instead, we focus on the partial observability introduced by the internal state governing the behavior of other drivers. We consider four cars for observation space: the front neighbor of the ego vehicle, the vehicle right behind the merge point, the rear neighbor and front neighbor of the projection of the ego vehicle on the main lane. The ego vehicle can observe the longitudinal position and velocity of these four vehicles perfectly if they are in the field of view. The longitudinal position, speed, and acceleration of the ego car are observed. In addition, we compare the cases where the internal state is directly observable by the ego vehicle or not leading to two or three dimensions per neighbor cars. The total dimension of the observation space is 15 when the internal states are observed and 12 otherwise. We will refer to those two cases as RL with full observation (FO), that can observe the internal state, and standard RL that only observes positions and velocities. In addition, it can observe its own state (three dimensions). This observation vector is used as input to our RL agent. Figure 4 shows the learning loop with details about states and flow of computation.
3) Actions : The action space is discrete with 7 possible actions at each time step. The ego vehicle controls its motion by applying a change in acceleration. At each time step, the acceleration is updated:

$$a_t = a_{t+1} + \nabla a$$

where θa is the action in the set { -1 m/s^2, -0.5 m/s^2, 0 m/s^2; 0.5 m/s^2, 1 m/s^2 }. The agent can also apply a hard braking action and releasing action which instantaneously set the acceleration to -4 m/s^2 and 0 m/s^2 respectively.
4) Reward : The agent receives a penalty of -1 for colliding with other traffic participants and receives a bonuse of 1 for reaching a goal position defined 50 m after the merge point. The time minimization is incentivized by the discount factor with value of 0.95, the sooner the goal is reached, the less the bonus is discounted.
5) Transition : Each vehicle follows a one dimensional point mass dynamics:

$$x_{t+1} = x_t + v_t \Delta t + \frac{1}{2} a_t \Delta t^2$$

$$v_{t+1} = v_t + a_t \Delta t$$

where x_t is the position of the vehicle at time t , v_t its velocity and at its acceleration.

B. Driver Model

To model the behavior of drivers on the main lane, we propose an extension of the Intelligent Driver Model (IDM). Our model controls the longitudinal acceleration of the vehicle on the main lane while taking into account merging vehicles. In addition to the IDM parameters, we introduce a cooperation level $c \in [0, 1]$ which is a scalar parameter controlling the reaction to the merging vehicle state. c = 1 represents a driver who slows down to yield to the merging vehicle if she predicts that the merging vehicle will arrive ahead of time. c = 0 represents a driver who completely ignores the merging vehicle until it traverses the merge point and follows standard IDM. We will refer to this model as Cooperative Intelligent Driver Model (C-IDM). C-IDM relies on estimating the time to reach the merge point (TTM) for the car on the main lane (TTM_a) and the car on the merge lane (TTM_b) to decide whether the merging vehicles should be considered or not. Once the time to merge for both vehicles is estimated, three cases are considered:

- If $TTM_b <$ c * TTM_a, the vehicle on the main lane follows IDM by considering the projection of the merging vehicle on the main lane as its front car.
- If $TTM_b \geq$ c * TTM_a, the vehicle on the main lane follows standard IDM.
- In the absence of a merging vehicle or far from the merging vehicle, the C-IDM driver follows the standard IDM.

Although this model might not represent precisely how human drivers behave, it provides us with a broad range of behaviors by adjusting the cooperation level. In this work we used a simple constant velocity model to estimate the time to merge. A more sophisticated prediction model can be used to have more realistic estimates of the TTM. Given the cooperation level, the driver has a deterministic behavior.

C. Inferring the Cooperation Level

Since the cooperation level cannot be directly measured, the ego car maintains a belief on the cooperation level of the observed drivers. At each time step, the belief is updated using a recursive Bayesian filter given the current observation of the environment. In this problem, the position and velocity of other drivers is assumed to be fully observable whereas the cooperation level is not always observed. This assumption of mixed observability can help us reduce the computational cost of the belief update. The agent only maintains a distribution over the cooperation level of observed drivers instead of estimating the full state of the environment. Our simple belief updater, is acting as if the cooperation level was binary although it can take a continuous value. The belief at time t is composed of the fully observable part of the state, o_t, and θ_i for $i = 1...n$, where n is the number of observed drivers. θ_t^i represents the probability that vehicle i has a cooperation level of 1. At time $t + 1$, the ego vehicle observes o_{t+1} and updates its belief on the cooperation level of vehicle i as follows:

$$\theta_{t+1}^i = \frac{Pr(o_{t+1}|o_t, c_i = 1)\theta_t^i}{Pr(o_{t+1}|o_t, c_i = 1)\theta_t^i + Pr(o_{t+1}|o_t, c_i = 0)(1 - \theta_t^i)}$$

This equation consists of simulating forward the previous state with the two possible hypothesis: $c_i = 1$ and $c_i = 0$ and comparing the outcome with the current observation. The probability of transitioning from o_t to o_{t+1} given the cooperation level, is computed by propagating the state forward using the proposed transition model and assuming a Gaussian distribution centered around the predicted value with a standard deviation of 1 m for the positions and 1 m/s for velocities. Without this addition of noise in the transition model, the belief would converge after one observation to $\theta^i = \theta$ or $\theta^i = 1$ since the two models would be perfectly distinguishable.

The focus of this work is not to develop an efficient driver state predictor but rather discuss how this information can be used by an RL agent. Future work might consider more complex filtering techniques such as multi-hypothesis filters, interacting multiple models, or data-driven approaches to estimate the driver cooperation level from observation [20]. An additional extension is to consider continuous values of the cooperation level since it is supported by the C-IDM model.

D. Belief State Reinforcement Learning

Standard reinforcement learning techniques assume that the underlying environment is an MDP. Latent states such as driver behavior characteristics are not explicitly inferred during training. Although memoryless policies can be efficient, reasoning about latent states can often lead to a significant improvement [8]. In the merging scenario, knowing which drivers are more cooperative can help the agent take better decisions. It is expected that the ego vehicle will only try to merge in front of cooperative drivers. In order to learn such a behavior through reinforcement learning, we propose to use the belief state as input to the reinforcement learning policy. The resulting algorithm is very similar to the standard DQN algorithm applied to the belief MDP. A transition in the belief state MDP can be described as follows:

- At time step t, the agent has a belief b_{t-1} and receives an observation o_t
- The new belief b_t is computed.
- The agent takes action $a_t = argmax_a Q(b_t, a)$ The rest of the algorithm is identical to standard DQN. The input to the network is a vector of dimension 15:

$$b_t = [o_t, \theta_t^1, ..., \theta_t^n]$$

where n is the number of observed vehicle, and ot is the fully observable part of the state (information on position and velocity), and θ_t^i is the probability of driver i being cooperative. By feeding in the probability on the internal state, the agent can reason in the belief space rather than in the observation space. The resulting Q network is combined with a belief updater at test time to result in a policy robust to partial observability.

Table I
SIMPLE TABLE

Hyperparameters	Value
Neural network architecture	2 dense layers of 64 and 32 nodes
Training steps	3 × 106
Activation functions	Rectified linear units
Replay buffer size	400 k experience samples
Target network update frequency	5 k episodes
Discount factor	0.95
Optimizer	Adam
Learning rate	1 × 10-4
Exploration strategy	ϵ greedy
Exploration fraction	0.5
Final ϵ	0.01

IV. IMPLEMENTATION

This section highlights some important aspects of our implementation: the RL training procedure and the distribution of scenarios used during training. Further details can be found in our code base.

A. RL Training Procedure

We used a curriculum learning approach to train the agent by gradually increasing the traffic density. When training an RL agent in dense traffic directly, the policy converges to a suboptimal solution which is to stay still in the merge lane and does not leverage the cooperativeness of other drivers. Such a policy avoids collisions but fails at achieving the maneuver. To encourage exploratory actions, we first train an agent in an environment with sparser traffic (5 to 12 cars). An alternative to curriculum learning is to design the reward function to incentivize the learning agent to move forward at each time step. However, a more complex reward function often requires a lot of parameter tuning . We found the curriculum learning approach more practical in this work.

The parameters of the DQN algorithm are summarized in table I. It was implemented using the Flux.jl library [20].Training one policy took around 40 minutes on three million examples.

1) Initial State Distribution: To populate the initial state of the merging scenarios, we used a two step procedure. The first step consists of sampling the number of vehicles present from a desired range. We considered two ranges corresponding to different traffic conditions:

1) Mixed traffic: between 5 and 12 cars on the main lane. The agent will experience both sparse traffic scenarios and dense traffic scenarios
2) Dense traffic: between 10 and 14 cars on the main lane.

The main lane has a length of 150 m and the vehicles have a length of 4 m. In dense traffic situations, the gap between vehicles varies from less than 2 m to larger distances. Once the number of cars is decided, vehicles are randomly positioned on the main lane. The initial velocity of each vehicle is drawn from a Gaussian distribution of mean 5 m/s and a standard deviation of 1 m/s Finally, the desired velocity of each vehicle is drawn uniformly from the set: f4 m=s; 5 m=s; 6 m=sg and

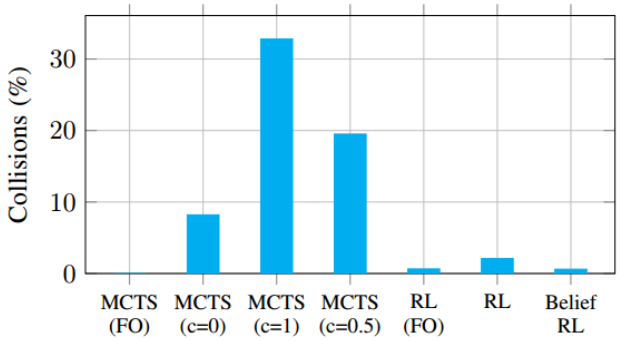

Figure 5. Collision Percentage

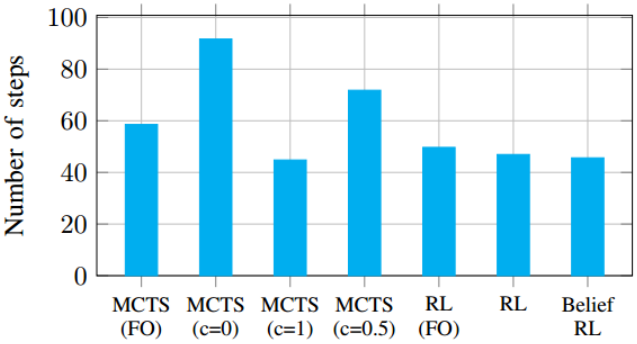

Figure 6. Number of Steps

their cooperation level is drawn uniformly in the interval [0; 1]. The desired velocity is used to parameterize the IDM part of C-IDM. The second step of the initialization consists of simulating the initial state for a burn-in time randomly chosen between 10 s and 20 s. During this burn-in time, the ego vehicle is not present and the vehicles on the main lane follow the C-IDM. The burn-in time allows the initial state to converge to a more realistic situation. This approach is loosely inspired by the work of Wheeler et al. [21]. Such procedure ensures that the learning agent experiences a variety of situations during training. The design of the training scenarios is critical to the performance of the RL agent and will determine the ability of the policy to generalize. The more variety it experiences during training, the more the policy can generalize.

V. RESULTS

Figure 5 illustrates the performance of the evaluate policy on a dense traffic scenario. We can see from the percentage of collisions that MCTS (FO) is the safest policy, followed by the belief RL approach. The three other MCTS policies had a lot of collisions (much larger than 1 %). The RL policy without access to information on the cooperation level had 2 % collisions at test time and the two other RL policies performed similarly with around 0.6 %collisions. Previous works has

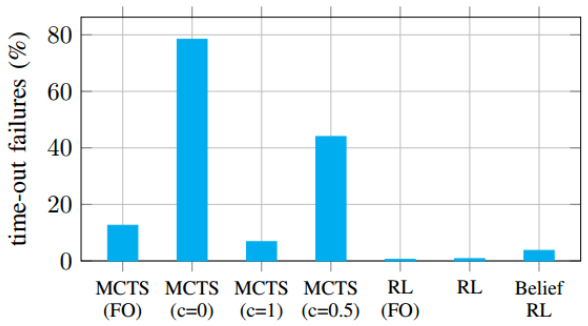

Figure 7. Time Out Failures

shown that only relying on deep RL is not sufficient to achieve safety [12]. The deployment of those policies would require the addition of a safety mechanism. It is important to notice that even though they did not have access to the full state, the RL and belief RL policy have much better safety performance than the MCTS approaches that did not have access to this information either.

Regarding the number of time steps to cross, we notice that the MCTS policy with c = 1 is the most efficient. The policy is biased towards taking more aggressive actions since it is assuming that every driver is cooperative. On the contrary, the MCTS policy assuming that no driver is cooperative (c = 0) has a very conservative behavior: it takes the longest time and has the largest number of time-out failures. As expected MCTS (c = 0:5) has a performance in between the previous two. The RL policies are more efficient than the MCTS (FO) policy and have an average time to cross close to the most aggressive MCTS policy. In addition, we can see that the RL policies have much fewer time-out failures than the MCTS policies. This last fact illustrates that they were able to successfully infer and leverage the information on the cooperation level. One can notice that the gap in performance between MCTS (FO) and the other MCTS is much larger than the gap between RL and RL (FO). A possible explanation is that the neural network approximation is able to capture the cooperation level inference task in the hidden layers. However, this implicit state estimation is less efficient than the explicit state estimation provided by combining the belief updater with the RL policy during training and execution. The belief RL policy has a similar safety level and takes, on average, a similar number of steps than the RL policy with perfect observation.

MCTS with full observation still presents a significant number of time-out failures. We believe that relaxing the computation constraint would have lead to better performance. Computation is generally the major bottleneck of online planning algorithms. Our experiments show that the performance can be closely matched using offline trained policies which take a very short time to execute online. However, the MCTS policy with full observation is safer than the deep RL equivalent. A direction for future work is to use the RL policy as a value estimator to guide the search in MCTS.

VI. Conclusion

Driving comprises of many tasks. Some tasks are easily performed by driver assist systems whereas some tasks are difficult especially when effect of other vehicles or dynamic nature is involved. Here we study the challenges in automatic merging with dense traffic and solve it using reinforcement learning. Using deep reinforcement learning the agent is trained to make sense of cooperation of other cars on the road. Reinforcement learning is closest to the method by which humans learn. And driving is often described as whatever humans do when they drive. This approach is quite promising. The accuracy of the individual components of our design has huge scope for improvement. Moreover, since we execute these components one after the other sequentially, the overall accuracy falls down, as the errors of each component get carry forwarded and affect the further components.

REFERENCES

[1] R. Sutton, A. G. Barto *Reinforcement Learning*.

[2] P. Trautman, A. Krause *Unfreezing the robot: Navigation in dense, interacting crowds*. International Conference on Intelligent Robots and Systems (IROS) 2010.

[3] E. Ward, N. Evestedt, D. Axehill, and J. Folkesson *Probabilistic Model for interaction aware planning in merge scenarios*. IEEE transactions on Intelligent Vehicles, vol. 2, no. 2, pp. 133-146, 2017.

[4] C. Hubmann, J. Schulz, G. Xu, D. Althoff, and C. Stiller *A belief state planner for interactive merge maneuvers in congested traffic* IEEE International Conference on Intelligent Transportation Systems (ITSC), 2018

[5] E. Schmerling, K. Leung, W. Vollprecht, and M. Pavone *Multimodal probabilistic model-based planning for humanrobot interaction*. IEEE International Conference on Robotics and Automation (ICRA), 2018.

[6] J. F. Fisac, E. Bronstein, E. Stefansson, D. Sadigh, S. S. Sastry, nd A. D. Dragan *Hierarchical game-theoretic planning for autonomous vehicles*. IEEE International Conference on Robotics and Automation (ICRA), 2019.

[7] S. Bansal, A. Cosgun, A. Nakhaei, and K. Fujimura *Collaborative planning for mixed-autonomy lane merging*. 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2018.

[8] Z. N. Sunberg, C. J. Ho, and M. J. Kochenderfer *The value of inferring the internal state of traffic participants for autonomous freeway driving*. American Control Conference (ACC), 2017.

[9] C. Dong, J. M. Dolan, and B. Litkouhi *ntention estimation for ramp merging control in autonomous driving (in review)*. IEEE Intelligent Vehicles Symposium (IV), 2017.

[10] D. Sadigh, S. Sastry, S. A. Seshia, and A. D. Dragan *Planning for autonomous cars that leverage effects on human actions*. Robotics: Science and Systems, 2016.

[11] P. Wang, C. Chan, and A. de La Fortelle *A reinforcement learning based approach for automated lane change maneuvers*. IEEE Intelligent Vehicles Symposium (IV), 2018.

[12] T. Tram, A. Jansson, R. Grönberg, M. Ali, and J. Sjöberg *Learning negotiating behavior between cars in intersections using deep q-learning*. IEEE International Conference on Intelligent Transportation Systems (ITSC), 2018.

[13] M. Bouton, A. Nakhaei, K. Fujimura, and M. J. Kochenderfer *Safe reinforcement learning with scene decomposition for navigating complex urban environments*. IEEE Intelligent Vehicles Symposium (IV), 2019.

[14] M. J. Kochenderfer *Decision making under uncertainty: Theory and application*. MIT Press, 2015.

[15] J. Loch and S. P. Singh *Using eligibility traces to find the best memoryless policy in partially observable markov decision processes*. International Conference on Machine Learning (ICML), 1998.

[16] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. A. Riedmiller, A. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg, and D. Hassabis *Human-level control through deep reinforcement learning*. vol. 518, no. 7540, pp. 529–533, 2015.

[17] T. Schaul, J. Quan, I. Antonoglou, and D. Silver *Prioritized experience replay*. International Conference on Learning Representations, 2016.

[18] Y. Hu, A. Nakhaei, M. Tomizuka, and K. Fujimura *Interaction-aware decision making with adaptive strategies under merging scenarios*. ArXiv preprint arXiv:1904.06025, 2019.

[19] M. Treiber, A. Hennecke, and D. Helbing *Congested traffic states in empirical observations and microscopic simulations*. Physical review E, vol. 62, no. 2, p. 1805, 2000.

[20] S. Thrun, W. Burgard, and D. Fox *Probabilistic robotics*. MIT press, 2005.

[21] M. Innes *Flux: Elegant machine learning with julia*. Journal of Open Source Software, 2018.

[22] T. A. Wheeler, M. J. Kochenderfer, and P. Robbel *Initial scene configurations for highway traffic propagation*. IEEE International Conference on Intelligent Transportation Systems (ITSC), 2015.

[23] A. Couëtoux, J. Hoock, N. Sokolovska, O. Teytaud, and N. Bonnard *Continuous upper confidence trees*. Learning and Intelligent Optimization (LION), 2011.