2018 IEEE International Conference on Robotics and Automation (ICRA)
May 21-25, 2018, Brisbane, Australia

Navigating Occluded Intersections with Autonomous Vehicles using Deep Reinforcement Learning

David Isele[1,3], Reza Rahimi[2], Akansel Cosgun[3], Kaushik Subramanian[4] and Kikuo Fujimura[3]

Abstract— **Providing an efficient strategy to navigate safely through unsignaled intersections is a difficult task that requires determining the intent of other drivers. We explore the effectiveness of Deep Reinforcement Learning to handle intersection problems. Using recent advances in Deep RL, we are able to learn policies that surpass the performance of a commonly-used heuristic approach in several metrics including task completion time and goal success rate and have limited ability to generalize. We then explore a system's ability to learn active sensing behaviors to enable navigating safely in the case of occlusions. Our analysis, provides insight into the intersection handling problem, the solutions learned by the network point out several shortcomings of current rule-based methods, and the failures of our current deep reinforcement learning system point to future research directions.**

I. INTRODUCTION

One of the most challenging problems for autonomous vehicles is to handle unsignaled intersections in urban environments. To successfully navigate through an intersection, it is necessary to understand vehicle dynamics, interpret the intent of other drivers, resolve the blind regions in case of occlusion, and behave predictably so that other drivers can respond appropriately. Learning this behavior requires optimizing multiple conflicting objectives including safety, efficiency, and minimizing the disruption of traffic. The ability to perform optimally at traffic junctions can both extend the abilities of autonomous agents and increase safety through driver assistance when a human driver is in control.

Several strategies have already been applied to intersection handling, including cooperative [1] and heuristic [2] approaches. Cooperative approaches require vehicle-to-vehicle communication and thus are not scalable to general intersection handling. The current state of the art is a rule-based method based on time-to-collision (TTC) [3], [4], which is a widely used heuristic as a safety indicator in the automotive industry [5]. Variants of the TTC approach have been used for autonomous driving [6] and the DARPA urban challenge, where hand engineered hierarchical state machines were a popular approach to handle intersections [7], [8]. TTC is currently the method we employ on our autonomous vehicle [9].

While TTC has many benefits - it is relatively reliable, generates behavior that is easy to interpret, and can be tuned to reach a high level of safety - it also has limitations. First, the TTC models assume constant velocity, which ignores nearly all information concerning driver intent. This is problematic: in the DARPA Urban Challenge, one reason behind a collision between two autonomous cars was "failure to anticipate vehicle intent" [10]. Even with higher order dynamics, the often-unpredictable behavior of human drivers complicates the use of rule-based algorithms. Second, in many cases an agent using TTC is overly cautious, creating unnecessary delays. Third, TTC methods assume full knowledge of their surroundings limiting their ability to handle occlusions. While special behaviors can be coded to address specific cases, this requires an engineer to identify and manage a large possible number of scenarios. These reasons motivate our investigation of machine-learning based approaches for intersection handling in autonomous vehicles.

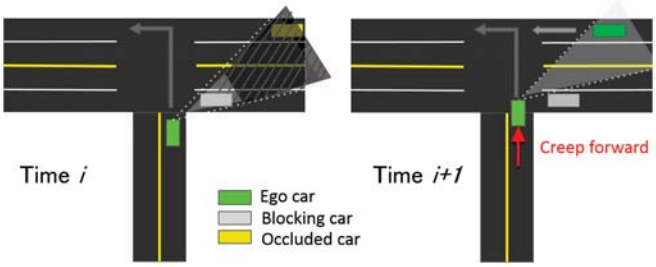


Fig. 1: Using creeping behavior to actively sense occluded obstacles. The objective is to determine the acceleration profile along the path while safely avoiding collisions.

Several machine learning based approaches have been used for the intersection handling, such as imitation learning, online planning, and offline learning. In imitation learning, the policy is learned from a human driver [11], however this policy does not offer a solution if the agent finds itself in a state that is not part of the training data. Online planners compute the best action to take by simulating the future states from the current time step. Online planners based on partially observable Monte Carlo Planning (POMCP) have been shown to handle intersections [12], but rely on the existence of an accurate generative model. Offline learning tackles the intersection problem, often by using Markov Decision Processes (MDP) in the back-end [13], [14].

In this paper, we explore the use of Deep Q-Networks (DQNs) [15], [16] for intersection handling with a focus on acting in the presence of partial knowledge. We view exploratory actions as a complementary approach to existing partially observable Deep RL methods which use histories to approach the problem of partial observability [17], [18]. Figure 1 shows an autonomous vehicle at an unsignaled intersection where an occlusion blocks the view of the road. In order to navigate safely, the agent must first take an exploratory action to better understand the environment.

The first contribution of this paper is demonstrating the

[1]The University of Pennsylvania, [2]The University of Virginia,
[3]Honda Research Institute,[4]The Georgia Institute of Technology

effectiveness of deep learning techniques on the intersection handling problem. We explore several network designs and present two we found to work well. The top performing system works by identifying the time to start accelerating. As neither TTC nor DQNs perform optimally in all cases, our analysis has interest beyond the specific choice of learner as we identify scenarios where TTC can be improved and suggests methods that could improve it.

The second contribution is the analysis of exploratory actions (creeping behaviors in specific) as a means of improving safety on autonomous vehicles. We show that occlusions create a need for exploratory actions and we show that deep reinforcement learning agents are able to discover these behaviors.

II. APPROACH

We view intersection handling as a reinforcement learning problem, and use a Deep Q-Network (DQN) to learn the state-action value Q-function.

A. Reinforcement Learning

In the reinforcement learning framework, at time t an agent in state s_t takes an action a_t according to the policy π. The agent transitions to the state s_{t+1}, and receives a reward r_t. The sequence of states, actions, and rewards is given as a trajectory $\tau = \{(\mathbf{s}_1, a_1, r_1), \ldots, (\mathbf{s}_T, a_T, r_T)\}$ over a horizon T.

A reinforcement learning task is typically formulated as a Markov Decision Process (MDP) $\langle \mathcal{S}, \mathcal{A}, P, \mathcal{R}, \gamma \rangle$, where $\mathcal{S}$ is the set of states, and $\mathcal{A}$ is the set of actions that the agent may execute. MDPs follow the Markov assumption that the probability of transitioning to a new state, given the current state and action, is independent of all previous states and actions $p(\mathbf{s}_{t+1}|\mathbf{s}_t, a_t, \ldots, \mathbf{s}_0, a_0) = p(\mathbf{s}_{t+1}|\mathbf{s}_t, a_t)$. The state transition probability $P : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0, 1]$ describes the systems dynamics, the reward function $\mathcal{R} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathbb{R}$ gives the real valued reward for a given time step, and $\gamma \in (0, 1]$ is a discount factor that adds preference for earlier rewards and provides stability in the case of infinite time horizons.

The goal of reinforcement learning is to choose the sequence of actions starting at time t that maximize the expected return, $R_t = \sum_{k=0}^{T} \gamma^t r_{t+k}$. We use Q-learning to perform this optimization.

B. Q-learning

In Q-learning [19], the action value function $Q^\pi(s, a)$ is the expected return $\mathbb{E}[R_t|s_t = s, a_t = a]$ for a state-action pair following a policy π. Given an optimal value function $Q^*(s, a)$ the optimal policy can be inferred by selecting the action with maximum value $\max_a Q^*(s, a)$ at every time step.

In Deep Q-learning [15], the optimal value function is approximated with a neural network $Q^*(s, a) \approx Q(s, a; \theta)$ with parameters θ. The action value function is learned by iteratively minimizing the error between the expected return and the state-action value predicted by the network

$$\mathcal{L}(\theta) = \Big(\mathbb{E}[R_t|s_t = s, a_t = a] - Q(s, a; \theta)\Big)^2 . \quad (1)$$

Since in practice the true return is not known, it is often approximated by the one-step return

$$\mathbb{E}[R_t|s_t = s, a_t = a] \approx r_t + \gamma \max_{a_{t+1}} Q(s_{t+1}, a_{t+1}; \theta) . \quad (2)$$

C. Action Representations

Autonomous vehicles use a suite of sensors, and allow for planning at multiple levels of abstraction. This allows for a wide variety of state and action representations. An exploratory search of representations showed that the selection of the representation had a significant impact on the agent's ability to learn. In this paper, we focus on three representations we found to work well.

a) Time-to-Go: In the Time-to-Go representation the desired path is provided to the agent and the agent determines the timing of departure through a sequence of actions to wait or go. Every *wait* action is followed by another wait or go decision, meaning every trajectory is a series of wait decisions terminating in a go decision, and the agent is not allowed to wait after the go action has been selected.

b) Sequential Actions: In the sequential action representation, the desired path is provided to the agent and the agent determines to accelerate, decelerate, or maintain constant velocity at every point in time along the desired path.

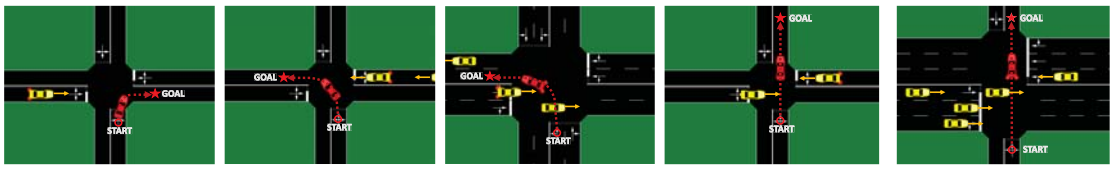
The sequential scenario allows for more complex behaviors: the agent could potential slow down half way through the intersection and wait for oncoming traffic to pass. The Time-to-Go representation more closely compares to TTC, placing importance on the time of departure. By analyzing the sequential action representation, we can observe if there is a benefit to allowing more complex behaviors. The Time-to-Go representation focuses on the departure time, allowing us to specifically probe how changes in departure time can affect performance.

c) Creep-and-Go: A third action set is constructed as a hybrid between sequential actions and Time-to-Go. Creep-and-Go involves three actions: wait, move forward slowly, and go. Like Time-to-Go, once a go action is selected the agent continues all the way through the intersection. This action configuration allows the agent to choose between moving up slowly and stopping before finally choosing a go action.

Creep-and-Go offers easier interpretability and ease of learning in a restricted action space like Time-to-Go, but makes exploratory actions available to the agent like in the sequential action system.

D. State Representation

A bird's eye view of space is discretized into a grid in Cartesian coordinates. Every car in the space is represented by its heading angle, velocity, and an indicator term that is 1 when a car occupies the position in the grid and 0 otherwise.



(a) *Right* (b) *Left* (c) *Left2* (d) *Forward* (e) *Challenge*

Fig. 2: Visualizations of intersection tasks used for our experiments.

Preliminary studies showed that real valued representations perform better, which we hypothesize is due to the reduced state space being easier to learn and the real values making it easier for the system to generalize.

In the occluded setting the representation is increased to include an indicator to identify if the cell is occluded, and the offset of the car's x and y offset to the pixel boundary to help assuage discretization affects.

III. EXPERIMENTS

We first train two different DQNs (Sequential Actions and Time-to-Go) on a variety of intersection scenarios and compare the performance against the heuristic Time-to-Collision (TTC) algorithm [3], [4], [9]. We then investigate a DQNs ability to learn exploratory behaviors in order to navigate occluded intersections.

A. Algorithm Comparison Study

Experiments were run using the Sumo simulator [20], which is an open source traffic simulation package. While we are presently restricting our reinforcement learning experiments to simulation for safety reasons, we verify that the representations generated from the simulator are qualitative similar to representations created from data collected from a real vehicle and have conducted initial investigations into transferring policies from simulation to a real vehicle [21]. We collected data from an autonomous vehicle in Mountain View, California, at an unsigned T-junction, similar to the *Left* scenario. A point cloud, obtained by a combination of six IBEO Lidar sensors, is first pre-processed to remove points that reside outside the road boundaries. A clustering method with hand-tuned geometric thresholds is used for vehicle detection. Each vehicle is tracked by a separate particle filter.

To simulate traffic in Sumo, users have control over the types of vehicles, road paths, vehicle density, and departure times. Traffic cars follow the Intelligent Driver Model (IDM) [22] to control their motion. In Sumo, randomness is simulated by varying the speed distribution of the vehicles and by using parameters that control driver imperfection (based on the Krauss stochastic driving model [23]). The simulator runs based on a predefined time interval which controls the length of every step. Our first set of experiments were to compare action representations against the TTC baseline. We ran experiments using five different intersection scenarios: *Right*, *Left*, *Left2*, *Forward* and a *Challenge*. Each of these scenarios is depicted in Figure 3. The *Right* scenario involves making a right turn, the *Forward* scenario involves crossing the intersection, the *Left* scenario involves making a left turn, the *Left2* scenario involves making a left turn across two lanes, and the *Challenge* scenario involves crossing a six-lane intersection with increased traffic density. The challenge scenario was intended to offer the sequential action model opportunity to benefit from more dynamic actions.

Each lane has a 45 mile per hour (20 m/s) maximum speed. The car begins from a stopped position. Each time step is equal to 0.2 seconds. The maximum number of steps per trial is capped at 100 steps which is equivalent to 20 seconds. The traffic density is set by the probability that a vehicle will be emitted randomly per second. We use 0.2 for all scenarios except the challenge scenario where it is set to 0.7.

We evaluate each method according to four metrics and run 10,000 trials of each scenario to collect our statistics. We use the following metrics:

- **Percentage of successes:** the percentage of the runs where the car successfully reached the goal. This metric takes into account both collisions and time-outs.
- **Percentage of collisions:** a measure of the safety of the method.
- **Average time:** how long it takes a successful trial to run to completion.
- **Average braking time:** the amount of time other cars in the simulator are braking. This can be seen as a measure of how disruptive the autonomous car is to traffic.

The networks architectures were optimized for sequential DQN and Time-to-Go DQN separately. For the sequential action DQN, space is represented as a 5×11 grid discretizing 0 to 20 meters in front of the car and ± 90 meters to the left and right of the car. Each spatial pixel, if occupied, contains the normalized real valued heading angles, velocity, and calculated time to collision. Exploratory studies found that this spatial representation outperformed higher dimensional representations with finer granularity. It was also found that real representations of the heading angle, velocity, and time to collision outperformed discretized versions. We hypothesize that this is because the real values allow for greater generalization.

For the Time-to-Go DQN, space is represented as an 18×26 grid. Unlike the sequential action DQN, this representation does not use the calculated time to collision for each car.

The sequential action network has 12 outputs correspond to three actions (accelerate, decelerate, maintain velocity) at four time scales (1, 2, 4, and 8 time steps) following the dynamic frame-skipping strategies of Lakshminarayanan et al. [24]. The final output layer for Time-to-Go uses five outputs: a single *go* action, and a *wait* action at four

time scales (1, 2, 4, and 8 time steps). Both networks are optimized using the RMSProp algorithm [25].

The sequential action network was slower to learn and performed considerably below the Time-to-Go. To put the performance of the sequential network on par with Time-to-Go an additional feature denoting the time to collision for each car was added to the state representation and the network was allowed to train longer. Each sequential action scenario was trained on one million simulations. Each Time-to-go scenario was trained on 250 thousand simulations.

For the reward, we used $+1$ for successfully navigating the intersection, -10 for a collision, and -0.01 step cost.

We additionally run experiments to test generalization. This is done by testing a policy trained in one scenario on other scenarios. Each policy is directly copied, and we do not do any additional training on the new scenarios. For generalization we used fewer training simulations (25,000) to prevent over-fitting.

B. Experiments with Occlusion

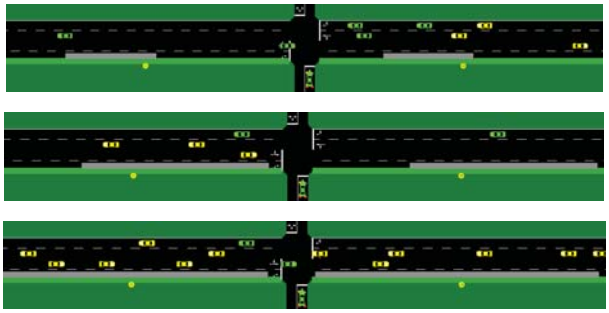


Fig. 3: Visualizations of occlusion experiments. Occluding objects are gray. Occluded cars are shown in yellow, and visible cars are shown in green.

Our second set of experiments investigates handling intersections with occlusions. We consider an intersection setup where the goal is to make a left turn across two lanes in the presence of occlusions. The closest lane contains objects that obstruct the vehicles view. The visible area is modeled using a ray tracing approach to find the occluded cells in the grid. We do not add the information of the cars that are in an occluded cell.

Occlusions are randomly selected to appear on the left, right or both sides of the intersection. The occlusions vary in dimensions with lengths$=\{30, 60, 90\}$ meters and widths$=\{0.5, 1, 2\}$ meters. Each occlusion is positioned a distance of $\{51, 61, 91\}$ meters from the left most boundary of that side. For our occlusion experiments we switch to a dueling network architecture [16] with prioritized replay [26] which we found less sensitive to tuning and we doubled the traffic density to make differences in approaches more apparent. To allow more time to creep forward and find an opening in the denser traffic we increased the time limit to 60 seconds or 300 simulation steps.

To emphasize the importance of localizing cars in the occlusion setting, non-ego cars are not allowed to decelerate. This removes much of the network's ability to infer driver intent that was present in the first experiment, but makes missing the presence of a car more likely to result in a collision. Because traffic cars do not brake for the ego-car we do not measure braking time in the occlusion experiments.

TABLE I: Comparison of Different Algorithms

Scenario	Metric	*TTC*	*DQN-Sequential*	*DQN-Time-to-Go*
Right	% Success	99.61	99.5	**99.96**
	% Collisions	**0.0**	0.47	0.04
	Avg. Time	6.46s	5.47s	**4.63s**
	Avg. Brake	**0.31s**	0.88s	0.45s
Left	% Success	99.7	**99.99**	**99.99**
	% Collisions	**0.0**	**0.0**	0.01
	Avg. Time	6.97s	5.26s	**5.24s**
	Avg. Brake	0.52s	**0.38s**	0.46s
Left2	% Success	99.42	99.79	**99.99**
	% Collisions	**0.0**	0.11	0.01
	Avg. Time	7.59s	7.13s	**5.40s**
	Avg. Brake	0.21s	0.22s	**0.20s**
Forward	% Success	**99.91**	99.76	99.78
	% Collisions	**0.0**	0.14	0.01
	Avg. Time	6.19s	**4.40s**	4.63s
	Avg. Brake	0.57s	0.61s	**0.48s**
Challenge	% Success	39.2	82.97	**98.46**
	% Collisions	**0.0**	1.37	0.84
	Avg. Time	12.55s	9.94s	**7.94s**
	Avg. Brake	**1.65s**	1.94s	1.98s

IV. RESULTS

Table I shows the results from our first set of experiments. To yield the best results for TTC, the threshold for each scenario is chosen as the lowest that achieve zero collisions. As a result, The TTC method did not have a collision in any of the scenarios. All other methods had non-zero collision rates for all scenarios, except DQN-Sequential for the *Left* scenario. Among DQN methods, DQN Time-to-Go had substantially lower collision rate than DQN-sequential. For scenarios except *Challenge*, DQN Time-to-Go had only 7 collisions in a total of 40,000 simulations.

We see that both DQN methods are substantially more efficient at reaching the goal than TTC. DQN Time-to-Go has the best task completion time in all scenarios, except *Forward*, where DQN-Sequential is faster. On average, DQN Time-to-Go was 28% faster at reaching goal than TTC, whereas DQN Sequential was 19% faster than TTC. Therefore, the DQN methods have potential to reduce traffic congestion due to their efficiency navigating intersections.

Braking time, a measure of how disruptive the car is to other traffic, is comparable for all methods. Because of this, we did not find conclusive evidence of DQN methods being more aggressive than TTC, despite DQN methods posting non-zero collision rates.

DQN Time-to-Go has the highest success rates in all experiments, except one, where its success rate is only marginally lower than TTC. Both DQN methods produce sound policies, evidenced by the ego car reaching the goal at least 99.5% of the time for all scenarios except *Challenge*.

An interesting result was that TTC did not reach the goal the majority of the time in the *Challenge* scenario, reaching the goal only 39.2% of the time, and posting a success rate only slightly more than Random (29.9%). For the same scenario, DQN Time-to-Go reached the goal 98.46% of the time, significantly outperforming other methods. We offer more insight on these results in Section IV-C.

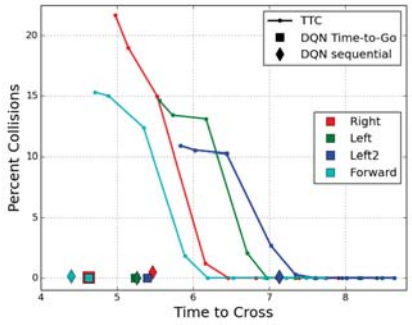


Fig. 4: Trade-off between the time to cross and collision rate as the TTC threshold is varied. Note that performance of the DQN dominates in every case. The challenge scenario is excluded for scale reasons, but the results are similar.

TABLE II: Comparison of Different Algorithms on Occlusion

Occlusion Length	Metric	TTC w creep	DQN-Time-to-Go	DQN-Creep	DQN-Sequential
30m	% Success	84.1	70.7	**92.8**	92.6
	% Collisions	**0.0**	29.2	7.1	7.3
	Avg. Time	28.6s	12.0s	11.7s	**9.6s**
60m	% Success	83.1	57.2	**92.9**	88.9
	% Collisions	**0.0**	36.65	7.1	11.05
	Avg. Time	28.9s	21.3s	16.0	**9.91s**
90m	% Success	87.6	26.3	**96.3**	86.2
	% Collisions	**0.0**	27.9	3.7	13.8
	Avg. Time	28.7s	30.8s	16.0s	**10.25s**

While the DQNs are substantially more efficient, they are seldom able to minimize the number of collisions as successfully as TTC. This is due to the fact that TTC has a tunable parameter that adjusts the safety margin and we tune TTC to the lowest threshold that gives zero collisions.

Comparing the DQN's performance against the TTC curve as we trade off speed vs. safety (Figure 4), we see that in every instance the DQN's performance dominates the performance of TTC. This suggests that it is possible to design an algorithm that has zero collision rate, but with better performance metrics than TTC.

A. Safety, Generalization, and Transfer

While the DQNs outperform TTC in terms of success rates, being able to achieve a zero percent collision rate is very important. The difficulty in shaping the reward to achieve zero collisions is a known problem with gradient based learning approaches [27]. Seamlessly integrating safety constraints into the learning process is the focus on ongoing research [28], [29]. However, unconstrained exploration is a useful tool for development - our experiments produced a policy where the autonomous vehicle aims to be very close behind a passing car in order to be as efficient as possible. This differed from the authors' natural driving, and appears to be a useful strategy for dense traffic. However, it is important to moderate the aggressiveness - applying the strategy too aggressively can create difficulty in interpreting intention and disrupt other drivers.

While we are examining safety constraints in our ongoing research, in this work we conducted an initial investigation into safety concerning how the system generalizes to out-of-sample data. To do this, we run the network trained in one scenario on every other scenario, directly copying the policy. The results are shown in Table III.

TABLE III: Transfer Performance for DQN Time-to-Go

Scenario	Training Method				
	Right	*Left*	*Left2*	*Forward*	*Challenge*
Right	**99.0**	98.5	81.1	98.1	74.1
Left	87.7	**96.3**	76.8	95.1	66.2
Left2	70.1	76.3	**91.7**	74.5	74.6
Forward	87.2	97.2	79.0	**97.6**	69.9
Challenge	58.7	60.7	72.3	62.4	**78.5**

For both the sequential and Time-to-Go DQNs we see that the networks trained on the *Left* scenario transfer better to the other single lane scenarios: *Right* and *Forward*. Similarly, the networks trained on the *Left2* scenario transferred well to the other multi-lane setting: *Challenge*. We believe this is largely due to the way the state is being represented. Notably, no method transfers well to all tasks. For a deeper investigation of how training on multiple tasks can increase a systems ability to generalize, we refers the readers to [30].

B. Occlusion

Table II shows the results of our occlusion experiments. A network using sequential actions can reach the point of full visibility fastest as therefore tends to be faster than the creeping approach, however the restricted action space does make creeping easier to learn and leads to a higher success rate. The creep forward behavior also brings the car closer to the goal when the intersection is blocked, as a result the methods without the creeping behavior (TTC and Time-to-Go) tend to take longer on average.

We see that TTC without creeping incorrectly assumes the road is clear and results in collisions. Even though TTC with creeping behavior has no collisions, it has a high percentage of timeouts. Notably the DQN is able to learn the specialized behavior and perform it more efficiently than TTC.

C. Qualitative Analysis

Comparing trials of the learned DQN networks and TTC, the DQN strategies take into account predictive behavior of the traffic. The DQNs can accurately predict that traffic in

distant lanes will have passed by the time the ego car arrives at the lane. Also the DQN driver can anticipate whether oncoming traffic will have sufficient time to brake or not. The few collisions seem to relate to discretization effects, where the car nearly misses the oncoming traffic.

In contrast, TTC does not leave until all cars have cleared its path. In addition, TTC leaves a sufficient safety margin for oncoming cars in distant lanes, since the same safety margin is used, the gap gets exaggerated in closer lanes. As a result, TTC often waits until the road is completely clear, missing many opportunities to cross.

We see that selecting the departure time offers sufficient opportunity to improve over TTC without the need to incorporate the added complexity of allowing for dynamic acceleration and deceleration behavior. This holds even for the *Challenge* scenario. The Time-to-Go DQN often chooses departures when other cars are either in or approaching the intersection, correctly predicting that they will be clear by the time the car reaches that position. However, using only departure time becomes ineffective once the systems encounters occlusions. Without the ability to move to a more favorable vantage point, both TTC and Time-to-Go regularly encounter collisions. This study identifies occlusions as a case where exploratory behaviors are needed and shows that DQNs are capable of learning them.

V. CONCLUSIONS

We showed a first system that uses Deep Q-Networks for the specific problem of intersection handling and show that it is capable of learning exploratory behaviors to more fully understand the scene. In addition to having better task efficiency and success rates than the rule based method, we identify occlusions as a fail case for rule based methods that lack hand-coded accommodations. While networks can learn both departure times and creeping behaviors, and have some ability to generalize to novel domains and out-of-sample data, they do occasionally result in collisions. Consequently more research is required to increase robustness.

REFERENCES

[1] M. R. Hafner, D. Cunningham, L. Caminiti, and D. Del Vecchio, "Cooperative collision avoidance at intersections: Algorithms and experiments," *IEEE Transactions on Intelligent Transportation Systems*, vol. 14, no. 3, pp. 1162–1175, 2013.

[2] J. Alonso, V. Milanés, J. Pérez, E. Onieva, C. González, and T. De Pedro, "Autonomous vehicle control systems for safe crossroads," *Transportation research part C: emerging technologies*, vol. 19, no. 6, pp. 1095–1110, 2011.

[3] M. M. Minderhoud and P. H. Bovy, "Extended time-to-collision measures for road traffic safety assessment," *Accident Analysis & Prevention*, vol. 33, no. 1, pp. 89–97, 2001.

[4] R. van der Horst and J. Hogema, *Time-to-collision and collision avoidance systems*. na, 1993.

[5] K. Vogel, "A comparison of headway and time to collision as safety indicators," *Accident analysis & prevention*, vol. 35, no. 3, pp. 427–433, 2003.

[6] D. Ferguson, C. Baker, M. Likhachev, and J. Dolan, "A reasoning framework for autonomous urban driving," in *Intelligent Vehicles Symposium, 2008 IEEE*. IEEE, 2008, pp. 775–780.

[7] S. Kammel, J. Ziegler, B. Pitzer, M. Werling, T. Gindele, D. Jagzent, J. Schöder, M. Thuy, M. Goebl, F. von Hundelshausen, *et al.*, "Team AnnieWays autonomous system for the DARPA urban challenge 2007," in *The DARPA Urban Challenge*. Springer, 2009, pp. 359–391.

[8] C. Urmson, J. Anhalt, D. Bagnell, C. Baker, R. Bittner, M. Clark, J. Dolan, D. Duggins, T. Galatali, C. Geyer, *et al.*, "Autonomous driving in urban environments: Boss and the urban challenge," *Journal of Field Robotics*, vol. 25, no. 8, pp. 425–466, 2008.

[9] A. Cosgun, L. Ma, J. Chiu, J. Huang, M. Demir, A. M. Anon, T. Lian, H. Tafish, and S. Al-Stouhi, "Towards full automated drive in urban environments: A demonstration in gomentum station, california," *IEEE Intelligent Vehicles Symposium (IV)*, 2017.

[10] L. Fletcher, S. Teller, E. Olson, D. Moore, Y. Kuwata, J. How, J. Leonard, I. Miller, M. Campbell, D. Huttenlocher, *et al.*, "The MIT–Cornell collision and why it happened," *Journal of Field Robotics*, vol. 25, no. 10, pp. 775–807, 2008.

[11] M. Bojarski, D. Del Testa, D. Dworakowski, B. Firner, B. Flepp, P. Goyal, L. D. Jackel, M. Monfort, U. Muller, J. Zhang, *et al.*, "End to end learning for self-driving cars," *arXiv:1604.07316*, 2016.

[12] M. Bouton, A. Cosgun, and M. J. Kochenderfer, "Belief state planning for navigating urban intersections," *IEEE Intelligent Vehicles Symposium (IV)*, 2017.

[13] W. Song, G. Xiong, and H. Chen, "Intention-aware autonomous driving decision-making in an uncontrolled intersection," *Mathematical Problems in Engineering*, vol. 2016, 2016.

[14] S. Brechtel, T. Gindele, and R. Dillmann, "Probabilistic decision-making under uncertainty for autonomous driving using continuous pomdps," in *Intelligent Transportation Systems (ITSC), 2014 IEEE 17th International Conference on*. IEEE, 2014, pp. 392–399.

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

[16] Z. Wang, T. Schaul, M. Hessel, H. Van Hasselt, M. Lanctot, and N. De Freitas, "Dueling network architectures for deep reinforcement learning," *arXiv:1511.06581*, 2015.

[17] M. Zhang, Z. McCarthy, C. Finn, S. Levine, and P. Abbeel, "Learning deep neural network policies with continuous memory states," in *Robotics and Automation (ICRA), 2016 IEEE International Conference on*. IEEE, 2016, pp. 520–527.

[18] N. Heess, J. J. Hunt, T. P. Lillicrap, and D. Silver, "Memory-based control with recurrent neural networks," *arXiv:1512.04455*, 2015.

[19] C. J. Watkins and P. Dayan, "Q-learning," *Machine learning*, vol. 8, no. 3-4, pp. 279–292, 1992.

[20] D. Krajzewicz, J. Erdmann, M. Behrisch, and L. Bieker, "Recent development and applications of SUMO–simulation of urban mobility," *International Journal on Advances in Systems and Measurements (IARIA)*, vol. 5, no. 3–4, 2012.

[21] D. Isele and A. Cosgun, "Transferring autonomous driving knowledge on simulated and real intersections," *International Conference on Machine Learning Workshop (ICML-WS)*, 2017.

[22] 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.

[23] S. Krauss, "Microscopic modeling of traffic flow: Investigation of collision free vehicle dynamics," Ph.D. dissertation, Deutsches Zentrum fuer Luft-und Raumfahrt, 1998.

[24] A. S. Lakshminarayanan, S. Sharma, and B. Ravindran, "Dynamic frame skip deep q network," *arXiv:1605.05365*, 2016.

[25] T. Tieleman and G. Hinton, "Lecture 6.5-rmsprop, coursera: Neural networks for machine learning," *University of Toronto, Tech. Rep*, 2012.

[26] T. Schaul, J. Quan, I. Antonoglou, and D. Silver, "Prioritized experience replay," *arXiv:1511.05952*, 2015.

[27] S. Shalev-Shwartz, S. Shammah, and A. Shashua, "Safe, multi-agent, reinforcement learning for autonomous driving," *arXiv preprint arXiv:1610.03295*, 2016.

[28] J. H. Gillula and C. J. Tomlin, "Reducing conservativeness in safety guarantees by learning disturbances online: iterated guaranteed safe online learning," *Robotics: Science and Systems VIII*, p. 81, 2013.

[29] M. Alshiekh, R. Bloem, R. Ehlers, B. Könighofer, S. Niekum, and U. Topcu, "Safe reinforcement learning via shielding," *AAAI Conference on Artificial Intelligence (AAAI)*, 2018.

[30] D. Isele and A. Cosgun, "Selective experience replay for lifelong learning," *AAAI Conference on Artificial Intelligence (AAAI)*, 2018.