2021 IEEE International Conference on Robotics and Automation (ICRA 2021)
May 31 - June 4, 2021, Xi'an, China

2021 IEEE International Conference on Robotics and Automation (ICRA) | 978-1-7281-9077-8/21/$31.00 ©2021 IEEE | DOI: 10.1109/ICRA48506.2021.9561148

MIDAS: Multi-agent Interaction-aware Decision-making with Adaptive Strategies for Urban Autonomous Navigation

Xiaoyi Chen[1] Pratik Chaudhari[2]

***Abstract*—Autonomous navigation in crowded, complex urban environments requires interacting with other agents on the road. A common solution to this problem is to use a prediction model to guess the likely future actions of other agents. While this is reasonable, it leads to overly conservative plans because it does not explicitly model the mutual influence of the actions of interacting agents. This paper builds a reinforcement learning-based method named MIDAS where an Ego agent learns to affect the control actions of other cars in urban driving scenarios. MIDAS uses an attention mechanism to handle an arbitrary number of other agents and includes a "driver-type" parameter to learn a single policy that works across different planning objectives. We build a simulation environment that enables diverse interaction experiments with a large number of agents and develop methods for quantitatively studying the safety, efficiency, and interaction among vehicles. MIDAS is validated using extensive experiments and we show that it (i) can work across different road geometries, (ii) results in an adaptive Ego policy that can be tuned easily to satisfy different performance criteria, such as aggressive or cautious driving, (iii) is robust to changes in the driving policies of external agents, and (iv) is safer and more efficient than existing approaches to interaction-aware decision-making. Code available here.**

***Index Terms*—interaction-aware driving, urban autonomous navigation, reinforcement learning**

I. INTRODUCTION

Consider an autonomous vehicle (henceforth called the "Ego") that is turning left at an intersection: when sharing the road with other vehicles, Ego would predict the forward motion of the other cars and stop until it is deemed safe and legal to proceed. This is a reasonable approach when the right-of-way is clear, but it becomes inefficient [1] otherwise. This behavior is often exploited by human-driven vehicles†. We may mitigate this inefficiency by enabling Ego to communicate its intent to others through its actions, such as inching forward towards the intersection. This is powerful because it does not require additional infrastructure and enables other human drivers to reason about autonomous vehicles in the same way as they would reason about any other human-driven vehicle. This paper devises such driving strategies for autonomous vehicles that can influence the actions of other drivers, specifically other human drivers. Our contributions are as follows.

1. Modeling aggression of external agents and the Ego to drive safely and efficiently in interactive environments under partial observations. We model the interaction problem as a partially-observable Markov decision process where Ego only observes the states of other agents in its vicinity and maximizes a reward that encourages it to reach a goal region in minimal time while avoiding collisions. Non-Ego agents drive with an Oracle policy that has full access to trajectories of nearby agents. A hidden "driver-type" variable allows the user to tune the policy to be aggressive or cautious.

2. Ability to handle an arbitrary number of external agents, some of which do not affect the optimal action. We parametrize Ego's policy using an attention-based architecture which can handle an unordered, arbitrary number of agents in the observation range. Attention allows Ego to only focus on agents that are relevant to decision-making. In contrast, current literature studies interaction among a fixed [2] or limited number of agents [3], [4].

3. Learning to interact without knowing the dynamics of external agents. We use off-policy reinforcement learning (RL) methods to learn the policy and include two innovations which reduce the variance of the action selection and stabilize the target in temporal-difference learning.

4. A platform for extensive and systematic interaction experiments. We build a simulation environment that enables carefully designed interaction experiments with a large number of agents in diverse road geometries. We devise a number of metrics that allow fine-grained reporting of safety and efficiency of Ego's policy. We can quantitatively study how Ego influences the actions of external agents in this environment. We perform extensive experiments in realistic scenarios to show that MIDAS (i) can work across different road geometries, (ii) results in an adaptive Ego policy that can be tuned to satisfy different performance criteria, (iii) is robust to changes in the driving policies of external agents, and (iv) is safer and more efficient than existing approaches to interaction-aware decision-making.

II. RELATED WORK

Intention-aware planning with known dynamics for agents can be formulated as a POMDP [5] and solved using point-based solvers [6]. The focus is typically on planning a safe trajectory and not on interaction [7]. For urban driving, one may assume road-safety rules and use game-theoretic formulations to model multi-agent decision making [8], [9]. It is popular to use a prediction model [10] or incorporate actions of other agents [3] for interactive, model-free driving using deep RL. These papers involve fewer and easier interactions than in our experiments. Merging is studied in [2], [11], using model predictive control for estimating the future horizon cost via exhaustive search. MIDAS is complementary to these approaches. Although we do not use a prediction model, it can be easily incorporated using model-based RL techniques.

Our adjustable driver type is similar to [12] where agents are encouraged to merge aggressively by overtaking others. The authors use self-play to train the policy. While the competition

[1] Work done while at the University of Pennsylvania. Currently at Nuro, Inc. Email: xiaoyich@alumni.upenn.edu
[2] Department of Electrical & Systems Engineering and the GRASP Laboratory at the University of Pennsylvania. Email: pratikac@seas.upenn.edu
†Link: The View from the Front Seat of the Google Self-Driving Car

approach is reasonable for highway merging, the is likely to result in high collision rates in busy intersections such as ours.

[13] uses inverse RL to learn the cost function of human drivers and uses it to influence other drivers. This method is computationally expensive and is limited to simple interactions. It can be improved by separating long-term and near-time planning at the cost of optimality [14]. Our attention-based model can scale to interaction with a large number of agents.

Learning-based approaches use featurization to tackle different lane geometries [4], roundabouts [15], or use simplified sequential action representations [16]. The observation vector is a concatenation of the states of other agents [2], spatial pooling that aggregates past trajectories of each agent [17], or birds-eye-view rasterization [18], [19]. In contrast, the set-transformer [20] in MIDAS is an easy, automatic way to encode variable-sized observation vectors. Our work is closest to "Social Attention" in [21] which learns to influence other agents based on road priority and demonstrates results on a limited set of road geometries. MIDAS compares favorably to this method in our experiments in Sec. V.

III. PROBLEM FORMULATION

Consider n agents with the state of the k^{th} vehicle at time $t \in \mathbb{Z}_{\geq 0}$ denoted by $x_t^k \in \mathbb{R}^d$. Denote the combined state of all agents by $x_t^{\text{all}} := \left[x_t^1, \ldots, x_t^n\right]$. The control input for the k^{th} agent and the control input of the combined system are denoted by u_t^k and $u_t^{\text{all}} = [u_t, \ldots, u_t^n]$ respectively. We will consider a deterministic discrete-time system $x_{t+1}^{\text{all}} = f(x_t^{\text{all}}, u_t^{\text{all}});\ x_0^{\text{all}} \sim p_0$ where the initial state x_0^{all} is drawn from a distribution p_0. The Ego agent has index 1. To simplify notation, we will denote Ego state, control and driver-type by x_t, u_t and β, respectively. The notation x_t^{-e} and β^{-e} refer to the combined state and driver-type of all agents other than Ego.

Driver-type and observation model. Each agent possesses a real-valued parameter $\beta^k \in [-1, 1]$ that models its "driver-type". A large value of β^k indicates an aggressive agent and a small value indicates that the agent is inclined to wait for others before making progress. An agent's driver-type is hidden to other agents. At each time t, agent k has access to observations $o_t^k = \left\{x_t^i :\ d(x_t^k, x_t^i) \leq D, i = 1, \ldots, n\right\}$ that consist of the states of all agents (including k) within some distance D. This model is a multi-agent Partially-Observable Markov Decision Process (POMDP): agents do not have access to the entire state of the problem due to a limited observation range, and they cannot observe the driver-type of others. Agent-specific goal locations x_g^k are sampled randomly using a goal distribution p_g. The reward function $r^k(x_t^{\text{all}}; x_g^k, \beta^k)$ encourages agents to reach their goal location in the shortest time while avoiding collisions with other agents. Ego's goal state is denoted by x_g. Collision is defined as two agents coming within some distance $\delta_{\text{collision}}$ of each other, assuming a uniform rectangular geometry for all agents. The deterministic control policy for each agent is $\pi^k(o_t^k;\ \beta^k)$. Given policies of the other agents π^{-e} and the initial state x_0, Ego maximizes

$$\mathop{\mathbb{E}}_{x_0^{\text{all}} \sim p_0, x_g^{\text{all}} \sim p_g, \beta^{-e}} \left[\sum_{t=0}^{\infty} \gamma^t r(x_t^{\text{all}}; x_g, \beta) \;\middle|\; x_0, \pi^{-e} \right]. \tag{1}$$

to obtain a policy $\pi^*(\beta)$ which is a function of its driver-type β. The control action $u_t = \pi(o_t; \beta)$. Ego's policy can thus be easily evaluated for a different value of β. The constant $\gamma < 1$ is a discount factor.

Simplifications. We are interested in $n \in [0, 25]$ agents and the problem formulation above is a large decentralized POMDP which is known to be intractable [22]. We make the following simplifications. (i) We include waypoints as part of the observation vector o_t^k. This enables the policy π^k to focus on handling interactions instead of spending sample complexity and model capacity on learning motion-planning behaviors. A shortest-path algorithm that finds the trajectory from the agent state x_t^k to its goal state x_g^k, without considering other agents on the road as obstacles, is used to compute these waypoints. (ii) Control actions of all agents are $u_t^k \in \{0, 1\}$, corresponding to stop and go. (iii) All non-Ego agents use the same policy π^{-e}. This policy, called the Oracle, is a user-designed, human-explainable deterministic policy (details in Sec. V-A) and serves as a benchmark for learned policies. These simplifications reduce our problem formulation to that of a standard POMDP, albeit with unknown dynamics, where only Ego's policy is learned.

IV. THE MIDAS APPROACH

A. Off-policy Training

Define the action-value function

$$q(o, u; \beta) = \mathop{\mathbb{E}}_{x_g^{\text{all}} \sim p_g, \beta^{-e}} \left[\sum_{t=0}^{\infty} \gamma^t r(x_t^{\text{all}};\ x_g, \beta) \;\middle|\; u_t = \pi(o_t; \beta) \right] \tag{2}$$

as the expected reward obtained using the policy π after starting from the initial state $x = x_0$ and action $u = u_0$. Off-policy methods are a popular technique to estimate this value function and learn the optimal policy by minimizing the Bellman error, also called the 1-step temporal difference (TD) error,

$$\text{TD} := q(o_t, u_t) - r(x_t^{\text{all}}; x_g, \beta) - \gamma\ q(o_{t+1}, \pi(o_{t+1}; \beta)) \tag{3}$$

over a dataset $\mathcal{D} = \{(o_t, u_t, r_t, o_{t+1})\}_{t=0,\ldots,}$ collected with a (behavior) policy that may be different from π. If the value function q_θ is parameterized using a function approximator, say a deep neural network with parameters θ, the TD-error is a function of these parameters θ. Further, if the control-space is discrete (as is the case for us) we can set $\pi(o_t; \beta) := \text{argmax}_{u'}\, q_\theta(o_t, u')$ and learn the optimal value function by performing stochastic gradient descent to minimize the average TD^2 error over the dataset. This objective can be seen as the regression error of the first term $q_\theta(o_t, u_t)$ in (3) against the sum of the other two terms which are together called the *target*. This forms the basis for the well-known DQN algorithm [23]. Off-policy methods are superior to others such as policy-gradient methods because they reuse the dataset $\mathcal{D}$ while learning q_θ, but implementing them in practice comes with a few caveats.

B. Tricks of the trade in off-policy RL

The TD-error can be zero even if the value function is not accurate because the Bellman operator is only a contraction in the ℓ_∞ norm, not the ℓ_2 norm [24]. This leads to instabilities during training, but they can be mitigated by several tricks that replace the value function $q_\theta(o_{t+1}, \text{argmax}_{u'}\, q_\theta(o_{t+1}, u'))$ used in (3). A popular scheme is to use time-lagged version of the

parameters, i.e., use $\max_{u'} q_{\theta_{\text{lag}}}(o_{t+1}, u')$ [23]. Over-estimation bias of the value function estimate [25] is countered by using two (or more) copies of the parameters $\{q_{\theta^i}, i = 1, 2\}$ along with time-lagged versions for both and using the minimum of these two

$$\min_{i=1,2} \left\{ q_{\theta^i_{\text{lag}}}(o_{t+1}, \operatorname*{argmax}_{u'} q_{\theta^i_{\text{lag}}}(o_{t+1}, u')) \right\}$$

to compute the target. Parameters of both copies are updated using TD^2 minimization. A variant of this is Double Q Learning [26] which uses $q_{\theta_{\text{lag}}}(o_{t+1}, \operatorname{argmax}_{u'} q_{\theta}(o_{t+1}, u'))$. Notice that the target is computed with the time-lagged parameters but the action is computed with the *current, non-lagged* parameters. This ensures a fair evaluation of the value of the greedy policy given by the *current* parameters. The above techniques are typically used together in state-of-the-art off-policy algorithms.

C. Our improvements to off-policy RL

We introduce two variants which further improve the training stability. First, we mix the two parameter copies when using the Double Q Learning trick:

$$q_{\theta}(o_{t+1}, \operatorname*{argmax}_{u'} q_{\theta}(o_{t+1}, u')) = \min_{i \neq j} q_{\theta^i_{\text{lag}}}(o_{t+1}, \operatorname*{argmax}_{u'} q_{\theta^j}(o_{t+1}, u')). \quad (4)$$

This forces the first copy, via its time-lagged parameters, to be the evaluator for the second copy, and vice-versa. This leads to further variance reduction of the target in the TD objective. Second, we pick the action during *policy evaluation* using the average of the two copies $\pi(o_t; \beta) = \operatorname{argmax}_{u'} \frac{1}{2} \sum_{i=1}^{2} q_{\theta^i}(o_t, u')$ which reduces the variance of action selection. Observe that doing so does not invalidate the Bellman equation: if both the value function estimates are optimal, the action predicted by their average is also optimal.

D. Attention-based Policy Architecture

Our value function should be *permutation invariant*: ordering of the states in the observation vector o_t should not affect Ego's action. It should also be able to *handle an arbitrary number of other agents*. The optimal action only depends on the *actionable information* [27], i.e. agents whose states cause a change in Ego's action. Our architecture bakes in these properties.

Permutation-invariant input representation. [28] shows that a function $\phi(A)$ operating on a set A is invariant to permutation of the elements in A iff it can be decomposed as $\phi(A) = \rho(\sum_{a \in A} \varphi(a))$ for some functions φ and ρ. This is remarkable, because both φ, ρ can now be learnt. The summation (average pooling) enables $\phi(A)$ to handle sets with varying sizes. However, the summation assigns the same weight to all elements in the input. As we see in our experiments, a value function using this "DeepSet" architecture is likely to be distracted by agents that do not inform the optimal action.

Attention. An attention mechanism takes in elements of the set A and outputs a linear combination $z(a) = \sum_{a' \in A} \alpha_{a,a'} \varphi_v(a')$ where $\varphi_v(a)$ is the "value" embedding of a and weights $\alpha_{a,a'} = \langle \varphi(a), \varphi(a') \rangle$ compute the similarity of different elements. An attention module [29] generalizes this idea and uses $\alpha_{a,a'} = \langle \varphi_k(a), \varphi_q(a') \rangle$. Similarity is computed across a set of "keys" φ_k and "queries" φ_q, all of which are learned. This is an elegant way for the value function to learn key, query, value embeddings that pay more attention to parts of the input that are relevant to the output. E.g, [21] sets the key to be the Ego state encoding, which is compared to query and value embeddings generated from the observation vector.

Set-based attention [20] in MIDAS. The summation in attention throws away higher-order correlations. A better representation for set-valued inputs could use: (i) a *set-attention block* (SAB) (we use its efficient version *induced* SAB) which builds $\varphi_v(a)$ itself using self-attention; (ii)*pooling by multi-head attention* (PMA) that uses self-attention to aggregate features instead of simple summation; (iii) regularization techniques such as layer-normalization [30] and residual connections [31], and (iv) *multi-head attention* [29] that computes inner products independently across subspaces of the embedding.

Encoding Ego's driver-type. Similar to masking NLP [32], we want Ego's driver-type information to only affect its own state encoding in the observation vector. We use a two-layer perceptron with ReLU nonlinearities to embed the scalar variable β and add the output to Ego's state encoding (Fig. 3).

E. Designing the reward function

The reward function for Ego is designed to encourage it to make progress towards its x_g while minimizing the time taken and the collisions with the other agents on the road. It consists of: time-penalty for every timestep; reward for non-zero speed; timeout penalty if Ego can't reach goal within time limit $T_{\max}$; stalement penalty where all nearby agents including Ego are standstill, waiting for one of them to take initiative and break the tie; collision penalty; car-following penalty for following too close to the agent in front. All sub-rewards, except for the last one, depend linearly on β, with the form $w\beta + b$, where the weight w and the bias b are chosen to achieve good performance on downstream performance metrics *using a generic RL agent*: a multi-layer perceptron (MLP) trained using the standard time-lagged Q-learning algorithm without the two variants introduced in Sec. IV-A. We emphasize that designing Ego's reward this way is reasonable and indeed what an algorithm designer will do in practice [12].

$$r(x_t; x_g) = \begin{cases} 0.05\beta \quad 0.15 & \text{time penalty} \\ 0.5\beta + 1.5 & \text{if Ego speed is non-zero} \\ 5\beta \quad 20 & \text{timeout penalty if } t = T_{\max} \\ 0.5\beta \quad 1.5 & \text{stalemate penalty} \\ 5\beta \quad 45 & \text{collision penalty} \\ 2 + \frac{2}{1+e^{-d_{\text{follow}}}} & \text{if Ego within distance } \delta_{\text{follow}} \\ & \text{of the front car} \end{cases} \quad (5)$$

V. EXPERIMENTAL EVIDENCE

A. Evaluation Methodology

The outcome of a multi-agent planning problem under partial observations is inherently noisy. It is therefore important to build an evaluation suite which provides fine-grained understanding but also allows studying complex scenarios.

Simulation environment. Our simulation environment (Fig. 1) consists of 4 T-intersections, 4 turns and a roundabout. In each episode, agents are initialized with random locations and goals. Each agent has a random driver type β that determines its velocity: $v = 2.7\beta + 8.3$. The relationship

is based on the recommended turning velocities [33] at the map's turning radii. A timestep is 0.1s. Non-Ego agents run the Oracle planner. For any two agents $i, j, i \neq j$, let $\tau_{0,1}^{ij}$ be the time-to-collision (TTC) if i takes action zero (stops) and j moves forward along its waypoints henceforth. At every timestep, i compares $\tau_{0,1}^{ij}$ with $\tau_{1,0}^{ij}$ for all agents j within its observation range and stops if the former is larger. A tie is broken by giving priority to $\min(i, j)$, but any other deterministic mechanism can be used here. Oracle's observation vector contains the states of all agents within a fixed travel-distance along their paths. Non-Oracle agents do not have global information about other agents' paths. Their observation vector contains all agents within a *Euclidean* distance of 10m. All observations use an Ego-centric coordinate frame.

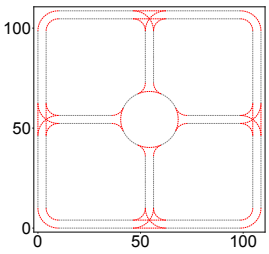


Fig. 1: **Environment Map**

Is the simulation environment too easy? Careful control our experiments at the right level of abstraction is crucial for a systematic study of interaction. Overly complex simulators, such as CARLA [34] with photorealistic rendering and SUMO [35] with city-scale maps, make it hard to draw precise conclusions about interactions. We ensure that our environment models key elements of the problem: non-centralized policies with adjustable aggressiveness, partial observations and human-explainable driving behavior. The Oracle policy is a non-trivial function of the TTC of all agents in the vicinity, which is determined by agent speeds, lane lines, stop/go decisions, etc. In other words, a "planner" is used to compute the TTC. Together with the tunable "driver-type" parameter hidden to others, Oracle policies are capable of complex interactions.

Datasets for systematic evaluation. We curate (i) *generic episodes* with random initial locations and goals, (ii) *collision episodes* where Ego will collide with other agents if it does not stop appropriately, and (iii) *interaction episodes* in which 2–3 agents arrive at a location simultaneously, where Ego cannot do well unless it negotiates with other agents, as illustrated in Fig. 2. We randomize over the agent quantities, driver-types, agent IDs, road geometries, and add small perturbations to arrival times. A mix of the three kinds of episodes are used for training.

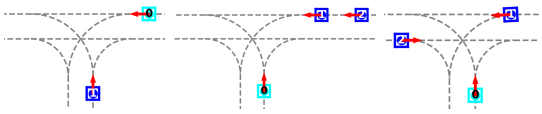

(a) Setting 1 (b) Setting 2 (c) Setting 3

Fig. 2: Simultaneous-arrival settings for generating the interaction set.

The test set uses a mix of generic and interaction episodes, so that it reflects general driving scenarios with an emphasis on interactions. We also use a hold-out test interaction set to evaluate performance in interactive scenarios. Curating the dataset in this way aids the reproducibility of results compared to only using random seeds, as is commonly done.

Evaluation metrics are (i) *time-to-finish* (TTF): the average episode length, and (ii) *collision, timeout and success rates*: the percentage of episodes that end with each status (the three add up to 1). We prioritize safety (low collision rate) over efficiency (low timeout rate and TTF). A safe and efficient driving policy should *keep collision rate low while maintaining a reasonable TTF*. A policy with low TTF and high collision rate (E.g. choosing to go all the time) is undesirable.

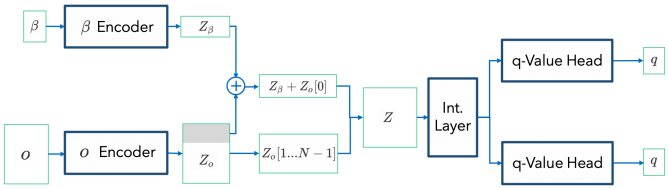


Fig. 3: **Schematic of MIDAS** o (Ego observation) encoder uses 2 layers of *induced* SAB [20] with 128 channels, 4 heads, 32 induced vectors. β (Ego driver type) encoder is a 2-layer perceptron with 64 and 128 channels and ReLU activation. The encoded β is added to the first row of the encoded o, forming Z. Intermediate Layer uses PMA with 128 channels, 4 heads, 2 seeds. The two identical q-value heads, both using SAB with 128 channels and 4 heads, each outputs q-value estimates of the two actions. MLP and Deep Set use comparable dimensions as MIDAS. We used the original implementation of SocialAttention [21]. Refer to the code for more details.

Baseline algorithms. We benchmark MIDAS against Oracle which is the performance upper bound, *Car Follower* which keeps a fixed minimum distance from the agent in front and is a performance lower bound, and learning-based models: MLP (a common learning baseline), Deep Set [28] (a popular architecture for prediction models [17]), and Social Attention [21] (an application of transformer for driving). Performances are reported over 4 random seeds.

Hyperparameters. $\gamma = 0.99$; policy updated at every episode-end for episode-length steps; θ_{lag} updated every 100 training steps with $\tau = 0.2$; replay buffer size 2×10^5; batch size 128; Adam optimizer learning rate 2×10^{-5}; first 500 training steps use ϵ-greedy, annealed exponentially from 1.0 to 0.01.

B. RESULTS

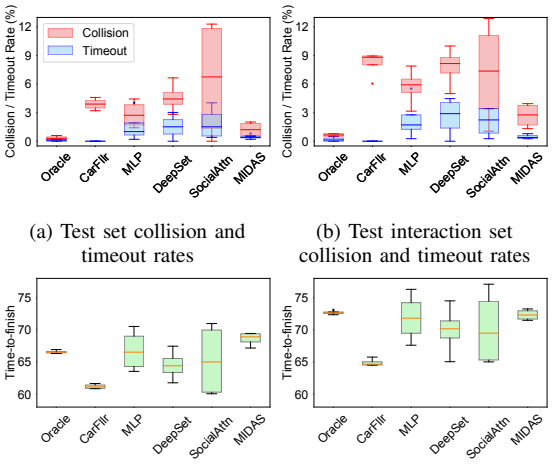
Table I and Fig. 4 report test performance on the test set and the test interaction set. We now discuss these results in the context of specific questions.

1. Is the learned model safe and efficient? Our experiments show that learning interactive driving policies is indeed beneficial: MLP and MIDAS both achieve lower collision rates than the naive rule-based Car-Follower in both general driving (Fig. 4a) and interactive driving (Fig. 4b) settings. MIDAS is safer than Car-Follower and other attention-based models while being as efficient as the Oracle. MIDAS has the lowest collision rate and timeout rate among all learned models. In terms of TTF (Fig. 4c, Fig. 4d), it is remarkable that MIDAS is similar to Oracle on the interaction set and only slightly slower (3.1%) than Oracle on the test set, even though MIDAS does not have all the information (long-time horizon, tie-breaking priority) that the Oracle has access to. MIDAS achieves efficiency while maintaining safety - even though other learned models have lower TTF, their mean collision rates are much higher.

2. Can the model generalize across different non-Ego driving policies? At test time, we add Bernoulli noise of probability 0.1 to other agents' actions to model the fact that

TABLE I: **Summary of empirical results**. Time-to-finish is average episode length within the set. Collision, timeout and success rates refer to the percentage of episodes in the set that end with the respective status. Lower collision rate indicates higher safety. Lower timeout rate and time-to-finish indicate higher efficiency. Higher success rate is better. Test set includes 250 *generic episodes* and 250 *interaction episodes*. Test interaction set includes 381 *interaction episodes*. The two sets don't overlap. We verified that most timeout cases for MIDAS are caused by Ego stepping into the TTC thresholds of other agents and then stopping for them, while the Oracle-driven agents also choose to wait, leading to a stalemate. In reality, stalemates like this are unlikely to happen because once Ego stops, the other drivers would likely break the tie.

Planner	**Test Set**				**Test Interaction Set**			
	Time-to-finish	**Collision (%)**	**Timeout (%)**	**Success (%)**	**Time-to-finish**	**Collision (%)**	**Timeout (%)**	**Success (%)**
Oracle	66.57 ± 0.23	0.35 ± 0.17	0.10 ± 0.10	99.55 ± 0.17	72.68 ± 0.27	0.66 ± 0.13	0.20 ± 0.22	99.14 ± 0.11
Car Follower	61.20 ± 0.32	3.90 ± 0.54	0.00 ± 0.00	96.10 ± 0.54	64.91 ± 0.51	8.16 ± 1.22	0.00 ± 0.00	91.84 ± 1.22
MLP	66.78 ± 2.85	2.82 ± 1.25	1.56 ± 1.47	95.61 ± 2.52	71.88 ± 3.30	5.72 ± 1.69	2.30 ± 1.95	91.97 ± 2.62
DeepSet [28]	**64.52 ± 2.04**	4.59 ± 1.37	1.51 ± 1.13	93.90 ± 1.37	**69.97 ± 3.35**	7.83 ± 1.81	2.57 ± 1.78	89.61 ± 1.15
SocialAttention [21]	65.27 ± 5.06	6.45 ± 5.59	1.86 ± 1.48	91.68 ± 4.36	70.27 ± 5.21	7.17 ± 4.75	2.04 ± 1.41	90.79 ± 3.53
MIDAS (Ours)	68.61 ± 0.92	**1.26 ± 0.66**	**0.45 ± 0.22**	**98.29 ± 0.54**	72.34 ± 0.74	**2.70 ± 1.14**	**0.46 ± 0.22**	**96.84 ± 1.05**



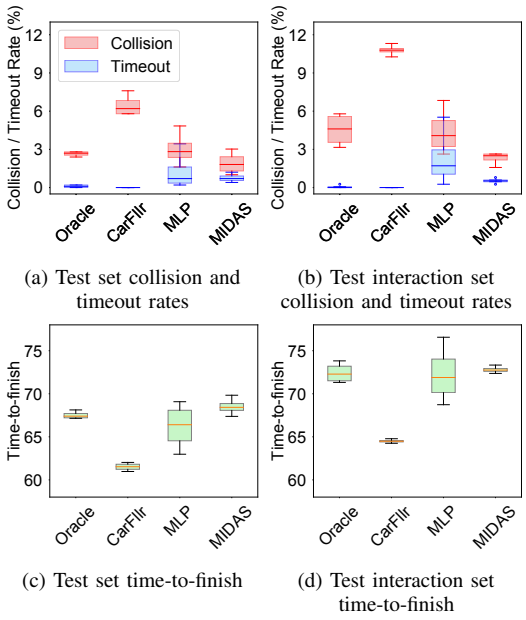
(a) Test set collision and timeout rates

(b) Test interaction set collision and timeout rates

(c) Test set time-to-finish

(d) Test interaction set time-to-finish

Fig. 4: **MIDAS is safer than Car-Follower and other learned planners, while being similarly efficient as Oracle.** Time-to-finish is average episode length within the set. Collision, timeout rates refer to the percentage of episodes that end with the respective status. CarFllr, SocialAttn refer to Car-Follower and Social Attention, respectively.



(a) Test set collision and timeout rates

(b) Test interaction set collision and timeout rates

(c) Test set time-to-finish

(d) Test interaction set time-to-finish

Fig. 5: **Test performance with action noise for external agents.** Oracle, Car-Follower collision rates greatly increase from Fig. 4 while performance of MLP and MIDAS is unchanged. MIDAS thus generalizes well to different driving policies for other agents.

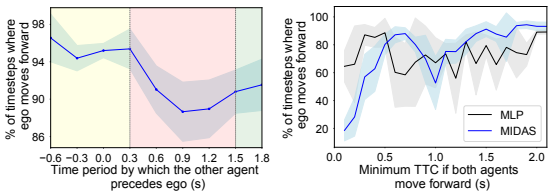
their driving policies may be different from each other and different from our gold-standard Oracle. As shown in Fig. 5, Oracle's collision rate is now larger, which can be directly attributed to conflicting actions taken by the agents during interaction. Car-Follower *becomes* more aggressive in this case and although its TTF is small, the collision rates are quite high. MLP performs about as well as the Oracle. In contrast to these, MIDAS has similar collision/timeout rates and TTF from Fig. 4 and outperforms Oracle in both driving settings (Fig. 5). This shows that MIDAS can generalize better to variations in the driving policies of other agents on the road.

3. Does MIDAS influence the actions of other agents? We curate test episodes where Ego and one other agent arrive simultaneously at the same location if they take their nominal actions. The arrival time of the other agent is perturbed to study how MIDAS changes its actions.

The X-axis in Fig. 6a is the time period (in seconds) by which the other agent *precedes* Ego. The Y-axis shows the percentage of timesteps where $u_t = 1$, i.e., Ego drives forward. The size of the agents is such that if the arrival times are within 0.3s of each other, either Ego or the other agent could go first. We see that Ego takes the initiative in more than 94% of the cases if it arrives earlier than 0.3s (yellow region), where the right-of-way is ambiguous. Ego is likely to stop if the other agent arrives earlier (red region). If the other agent arrives earlier by more than 1.5s (green region), the intersection is clear for Ego and the likelihood of proceeding forward goes back up. This experiment shows that in ambiguous situations, Ego attempts to drive forward, just like a human driver would in the absence of right-of-way rules. At the same time, Ego does stop if a collision is more likely.

Next we study Ego's behavior in safety-critical situations. Fig. 6b compares the percentage of timesteps where Ego goes forward (Y-axis) against the minimum TTC across all agents in the vicinity if both Ego and the other agent decide to go forward (X-axis), computed on the test set. For small TTC when collision is immediate, MIDAS is less likely to proceed

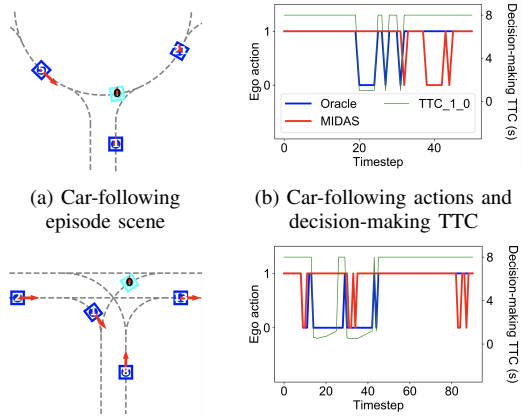
than MLP. This shows that MIDAS does not blindly drive forward in safety critical situations.



(a) (b)

Fig. 6: **Fig. 6a:** In ambiguous situations without clear right-of-way (yellow) or if the intersection clears before Ego arrives (green), MIDAS drives forward. But it stops more often if a collision is more likely (red). **Fig. 6b:** For small TTC when collision is immediate, MIDAS is less likely to proceed than MLP, indicating that MIDAS does not blindly drive forward in safety critical situations.

4. Does the model make decisions properly in highly interactive situations? We define highly interactive situations for agent i as those where $\tau_{1,0}^{ij}$ (green line in Fig. 7b, Fig. 7d) is lower than a threshold, indicating that there's at least one other agent very close by which forces i to stop. We run Oracle on the test set and *record what the learned model would have done if it were in the same situation* as the Oracle. In car-following (Fig. 7a), Oracle stops at a long distance away from the front vehicle at t=20, while blocking the merging traffic, but MIDAS chooses to move forward and stop closer to the front vehicle at t=38. In left-turn (Fig. 7c), Oracle waits for over 10 timestamps until there's a big clearance after the right-turning agent 1 before it proceeds, but MIDAS drives forward right after (at t=17). This shows that MIDAS drives more efficiently.



(a) Car-following episode scene

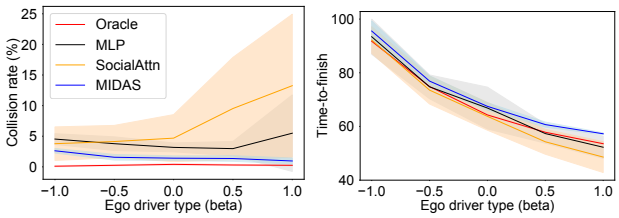
(b) Car-following actions and decision-making TTC

(c) Left-turn episode scene

(d) Left-turn actions and decision-making TTC

Fig. 7: **Typical episodes where MIDAS differs from Oracle.** Ego is cyan. In car-following, Oracle stops for front vehicle at a far distance and blocks traffic at an intersection, but MIDAS stops later. In left-turn, Oracle waits until there's a big clearance behind agent 1 before proceeding, but MIDAS proceeds right after. Both episodes show that MIDAS drives more efficiently. Plots show Oracle (blue) actions, MIDAS (red) actions and $\tau_{1,0}^{1j}$ (green).

5. Are MIDAS' strategies adaptive? How does model performance change with driver-type? At test time, we change Ego's driver type to $-1, -0.5, 0, 0.5, 1$ and run it on the test set with all other agents running the Oracle planner. Results shown in Fig. 8. MIDAS plans more efficiently as Ego driver type increases, shown by the decreasing time-to-finish (Fig. 8b), while maintaining an almost constant collision rate (Fig. 8a). Fig. 8a shows that MIDAS is consistently safer than all other learned models across different Ego driver types.



(a) (b)

Fig. 8: **MIDAS' strategy is adaptive.** Figs. 8a and 8b shows that MIDAS plans more efficiently but maintains constant collision rate as Ego driver-type changes. It's also safer than all other learned models. Deep Set and Car-Follower both perform worse than MIDAS and are not shown here for clarity. SocialAttn refers to SocialAttention.

6. MIDAS performs consistently across different driving scenes. Table II shows that the collision rate of MIDAS is consistently lower than other models across different road geometries. The collision/timeout rates also show only a small variation across intersection types.

TABLE II: **MIDAS performs consistently across different driving scenes.** Test result of Car-Follower, MLP, MIDAS across intersections.

Model	Intersection	Collision (%)	Timeout (%)	Success (%)
Car-Follower	T-intersection	4.20 ± 0.62	0.00 ± 0.00	95.80 ± 0.62
	Roundabout	2.88 ± 0.38	0.00 ± 0.00	97.12 ± 0.38
MLP	T-intersection	2.66 ± 1.00	2.01 ± 1.89	95.33 ± 2.71
	Roundabout	3.38 ± 2.33	0.00 ± 0.00	96.62 ± 2.33
MIDAS	T-intersection	1.23 ± 0.59	0.39 ± 0.13	98.38 ± 0.62
	Roundabout	1.35 ± 1.01	0.68 ± 0.75	97.97 ± 0.39

VI. DISCUSSION

Interaction in autonomous driving is difficult to analyze systematically across different scenarios. Our work is a step towards doing so using a carefully constructed simulation environment. This environment appears simple compared to others that are photo-realistic [34], [36] or have a large-scale map [35] but it allows building non-centralized complex interaction policies with adjustable aggressiveness and partial observation which is the goal of this paper. This level of abstraction is a conscious choice because complex simulators also require building good scene representations, making it hard to draw precise conclusions about interaction.

Our future work will focus on translating interaction-aware planning from simulation to reality. The main algorithmic challenge is to build intent-aware prediction models and perform probabilistic reasoning over their outcomes. To implement it on autonomous vehicles, we are cognizant that deep RL policies have a high performance variance in the real-world. We will focus on using MIDAS to provide a prior, which can be tuned to be more optimistic than a worst-case assumption, for systems built upon existing planning algorithms, such as rapidly-exploring random trees [37], [38], model predictive control [39], etc.

REFERENCES

[1] P. Trautman, J. Ma, R. M. Murray, and A. Krause, "Robot navigation in dense human crowds: Statistical models and experimental studies of human–robot cooperation," *The International Journal of Robotics Research*, vol. 34, no. 3, pp. 335–356, 2015.

[2] E. Schmerling, K. Leung, W. Vollprecht, and M. Pavone, "Multimodal probabilistic model-based planning for human-robot interaction," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2018, pp. 1–9.

[3] N. Rhinehart, R. McAllister, K. Kitani, and S. Levine, "Precog: Prediction conditioned on goals in visual multi-agent settings," in *Proceedings of the IEEE International Conference on Computer Vision*, 2019, pp. 2821–2830.

[4] C. Li and K. Czarnecki, "Urban driving with multi-objective deep reinforcement learning," in *Proceedings of the 18th International Conference on Autonomous Agents and MultiAgent Systems*. International Foundation for Autonomous Agents and Multiagent Systems, 2019, pp. 359–367.

[5] T. Bandyopadhyay, K. S. Won, E. Frazzoli, D. Hsu, W. S. Lee, and D. Rus, "Intention-aware motion planning," in *Algorithmic foundations of robotics X*. Springer, 2013, pp. 475–491.

[6] H. Bai, S. Cai, N. Ye, D. Hsu, and W. S. Lee, "Intention-aware online pomdp planning for autonomous driving in a crowd," in *2015 ieee international conference on robotics and automation (icra)*. IEEE, 2015, pp. 454–460.

[7] T. Wongpiromsarn, A. Ulusoy, C. Belta, E. Frazzoli, and D. Rus, "Incremental temporal logic synthesis of control policies for robots interacting with dynamic agents," in *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2012, pp. 229–236.

[8] P. Chaudhari, T. Wongpiromsarny, and E. Frazzoli, "Incremental minimum-violation control synthesis for robots interacting with external agents," in *2014 American Control Conference*. IEEE, 2014, pp. 1761–1768.

[9] M. Zhu, M. Otte, P. Chaudhari, and E. Frazzoli, "Game theoretic controller synthesis for multi-robot motion planning part i: Trajectory based algorithms," in *2014 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2014, pp. 1646–1651.

[10] M. Henaff, A. Canziani, and Y. LeCun, "Model-predictive policy learning with uncertainty regularization for driving in dense traffic," *arXiv preprint arXiv:1901.02705*, 2019.

[11] S. Bae, D. Saxena, A. Nakhaei, C. Choi, K. Fujimura, and S. Moura, "Cooperation-aware lane change maneuver in dense traffic based on model predictive control with recurrent neural network."

[12] 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.

[13] 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.

[14] J. F. Fisac, E. Bronstein, E. Stefansson, D. Sadigh, S. S. Sastry, and A. D. Dragan, "Hierarchical game-theoretic planning for autonomous vehicles," in *2019 International Conference on Robotics and Automation (ICRA)*. IEEE, 2019, pp. 9590–9596.

[15] J. Chen, B. Yuan, and M. Tomizuka, "Model-free deep reinforcement learning for urban autonomous driving," in *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*. IEEE, 2019, pp. 2765–2771.

[16] D. Isele, R. Rahimi, A. Cosgun, K. Subramanian, and K. Fujimura, "Navigating occluded intersections with autonomous vehicles using deep reinforcement learning," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2018, pp. 2034–2039.

[17] A. Gupta, J. Johnson, L. Fei-Fei, S. Savarese, and A. Alahi, "Social gan: Socially acceptable trajectories with generative adversarial networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 2255–2264.

[18] Y. Tang, "Towards learning multi-agent negotiations via self-play," in *Proceedings of the IEEE International Conference on Computer Vision Workshops*, 2019, pp. 0–0.

[19] M. Bansal, A. Krizhevsky, and A. Ogale, "Chauffeurnet: Learning to drive by imitating the best and synthesizing the worst," *arXiv preprint arXiv:1812.03079*, 2018.

[20] J. Lee, Y. Lee, J. Kim, A. R. Kosiorek, S. Choi, and Y. W. Teh, "Set transformer: A framework for attention-based permutation-invariant neural networks," *arXiv preprint arXiv:1810.00825*, 2018.

[21] E. Leurent and J. Mercat, "Social attention for autonomous decision-making in dense traffic," *arXiv preprint arXiv:1911.12250*, 2019.

[22] F. A. Oliehoek, C. Amato *et al.*, *A concise introduction to decentralized POMDPs*. Springer, 2016, vol. 1.

[23] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski *et al.*, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[24] D. P. Bertsekas, "Weighted sup-norm contractions in dynamic programming: A review and some new applications," *Dept. Elect. Eng. Comput. Sci., Massachusetts Inst. Technol., Cambridge, MA, USA, Tech. Rep. LIDS-P-2884*, 2012.

[25] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press, 2018.

[26] H. V. Hasselt, "Double q-learning," in *Advances in neural information processing systems*, 2010, pp. 2613–2621.

[27] S. Soatto, "Actionable information in vision," in *Machine learning for computer vision*. Springer, 2013, pp. 17–48.

[28] M. Zaheer, S. Kottur, S. Ravanbakhsh, B. Poczos, R. R. Salakhutdinov, and A. J. Smola, "Deep sets," in *Advances in neural information processing systems*, 2017, pp. 3391–3401.

[29] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," in *NIPS*, 2017.

[30] J. L. Ba, J. R. Kiros, and G. E. Hinton, "Layer normalization," *arXiv preprint arXiv:1607.06450*, 2016.

[31] K. He, X. Zhang, S. Ren, and J. Sun, "Identity mappings in deep residual networks," in *European conference on computer vision*. Springer, 2016, pp. 630–645.

[32] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," *arXiv preprint arXiv:1810.04805*, 2018.

[33] M. W. Hancock and B. Wright, "A policy on geometric design of highways and streets," *American Association of State Highway and Transportation Officials: Washington, DC, USA*, 2013.

[34] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun, "CARLA: An open urban driving simulator," in *Proceedings of the 1st Annual Conference on Robot Learning*, 2017, pp. 1–16.

[35] P. A. Lopez, M. Behrisch, L. Bieker-Walz, J. Erdmann, Y.-P. Flötteröd, R. Hilbrich, L. Lücken, J. Rummel, P. Wagner, and E. Wießner, "Microscopic traffic simulation using sumo," in *The 21st IEEE International Conference on Intelligent Transportation Systems*. IEEE, 2018. [Online]. Available: https://elib.dlr.de/124092/

[36] H. Zhao, A. Cui, S. A. Cullen, B. Paden, M. Laskey, and K. Goldberg, "Fluids: A first-order local urban intersection driving simulator," in *CASE*. IEEE, 2018.

[37] Y. Kuwata, G. A. Fiore, J. Teo, E. Frazzoli, and J. P. How, "Motion planning for urban driving using rrt," in *2008 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2008, pp. 1681–1686.

[38] A. Censi, K. Slutsky, T. Wongpiromsarn, D. Yershov, S. Pendleton, J. Fu, and E. Frazzoli, "Liability, ethics, and culture-aware behavior specification using rulebooks," in *2019 International Conference on Robotics and Automation (ICRA)*. IEEE, 2019, pp. 8536–8542.

[39] 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.