Nocturne: a scalable driving benchmark for bringing multi-agent learning one step closer to the real world

Eugene Vinitsky[*][†]
Meta AI, UC Berkeley
vinitsky.eugene@gmail.com

Nathan Lichtlé[*]
UC Berkeley
École des Ponts ParisTech
nathan.lichtle@gmail.com

Xiaomeng Yang[*]
Meta AI
yangxm@fb.com

Brandon Amos
Meta AI
bda@fb.com

Jakob Foerster
Univeristy of Oxford
jakob.foerster@eng.ox.ac.uk

Abstract

We introduce *Nocturne*, a new 2D driving simulator for investigating multi-agent coordination under partial observability. The focus of Nocturne is to enable research into inference and theory of mind in real-world multi-agent settings without the computational overhead of computer vision and feature extraction from images. Agents in this simulator only observe an obstructed view of the scene, mimicking human visual sensing constraints. Unlike existing benchmarks that are bottlenecked by rendering human-like observations directly using a camera input, Nocturne uses efficient intersection methods to compute a vectorized set of visible features in a C++ back-end, allowing the simulator to run at 2000+ steps-per-second. Using open-source trajectory and map data, we construct a simulator to load and replay arbitrary trajectories and scenes from real-world driving data. Using this environment, we benchmark reinforcement-learning and imitation-learning agents and demonstrate that the agents are quite far from human-level coordination ability and deviate significantly from the expert trajectories.

1 Introduction

This paper presents *Nocturne*, a new simulator and benchmark for multi-agent driving under human-like sensor uncertainty that is intended to aid the process of studying real-world multi-agent coordination and learning. Instead of combining the challenges of coordination with feature extraction from images, Nocturne is a 2D simulator that generates vector representations of the set of objects and road points that would be visible to an idealized human driver (see Fig. 1 for an example) and supports head-tilt to acquire additional information about blind spots. In contrast to driving benchmarks that achieve partial-observability by using a camera input, Nocturne uses efficient visibility-checking methods and a C++ back-end to construct observations and step the dynamics of a single agent at 2000 to 4000 steps per second. This speed is key to its use in multi-agent learning settings where frequently billions of environment interactions are needed to learn expert agents [2, 18]. In contrast to many existing multi-agent learning benchmarks, Nocturne is neither zero-sum nor fully-cooperative but mixed-motive, combining the challenges of coordination and cooperation.

Crucially, Nocturne is not just a simulator. Instead, it is built upon open-source driving data and features a diverse set of real-world scenes (see Fig. 2) that probe the ability of agents to safely navigate and coordinate in complex scenes such as intersections, roundabouts, parking lots, and highways. We

[*] **These authors contributed equally to this work.**
[†] Corresponding Author

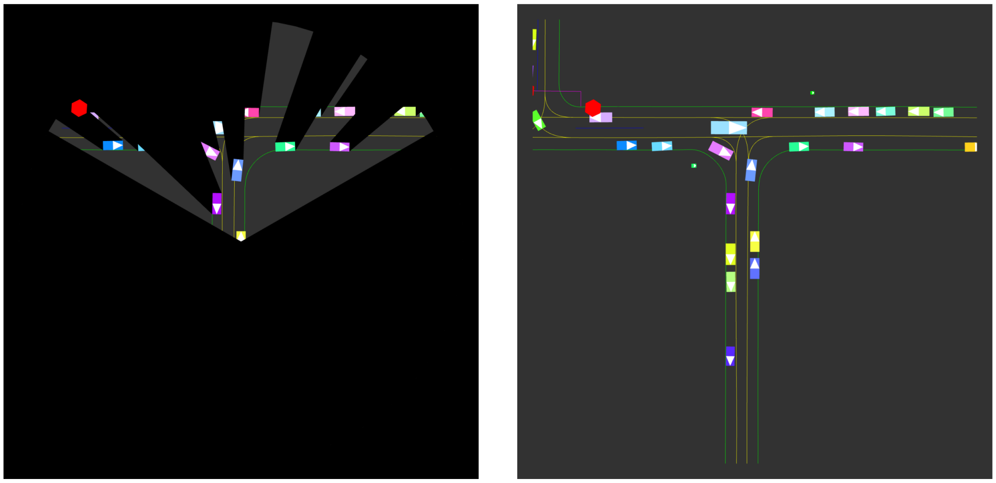


Figure 1: A visual depiction of the obstruction model used to represent objects that are visible to the agents. (Left) Obstructed view with the viewing yellow agent in the center of the cone. (Right) Original scene centered on the cone agent with an unobstructed view.

use this data as a source of experts for imitation, to flexibly vary the number of controlled agents in the scene, or as a train-test split for validating the generalization ability of a human-driver model.

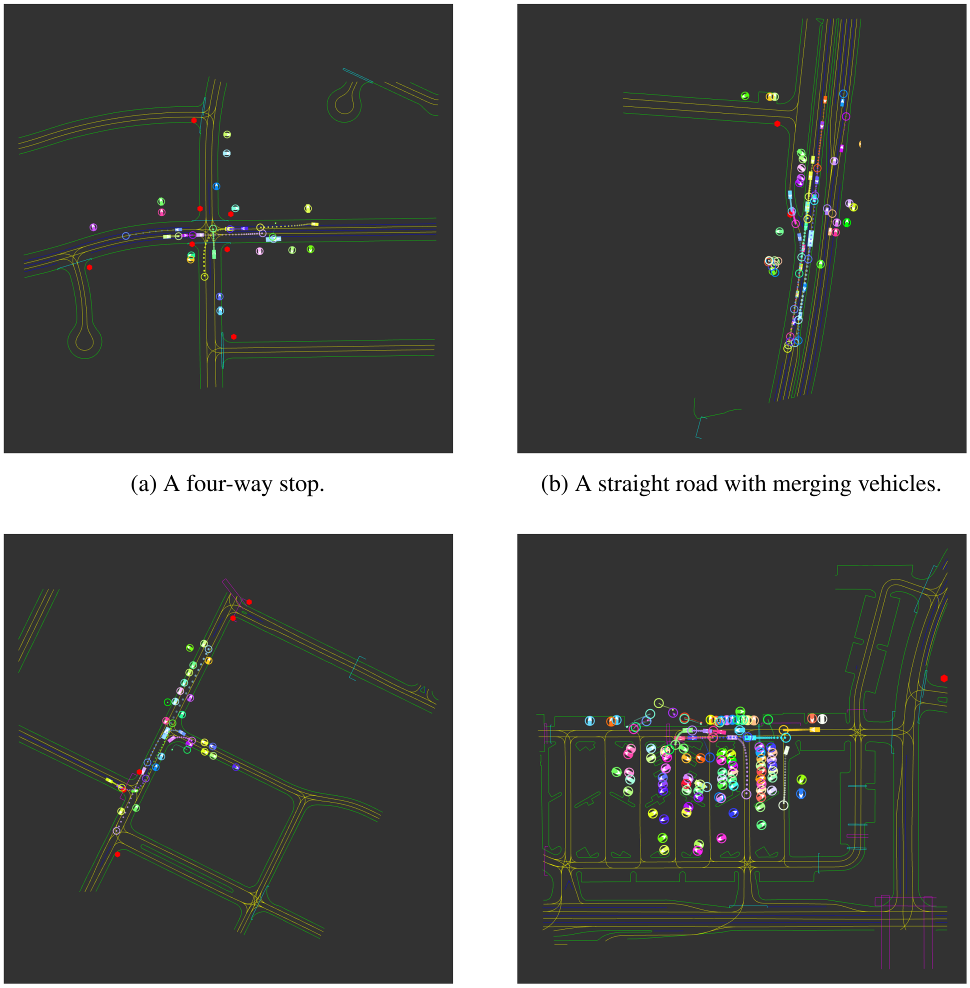
The challenge we propose is to learn (or otherwise design) policies that achieve the same set of final states as human experts (throughout this work we will call this the *goal rate* and the final state the *goal position*) while achieving a 0% collision rate. Achieving a high goal rate alongside a negligible collision rate is challenging for any policy design scheme (both learning and non-learning) due to the combination of the high-dimensional state space, the partial observability of the scene, the large number of interacting agents, and the decentralization of the policies at test time. The secondary challenge in Nocturne is to find policies that closely mimic human behavior at the trajectory level. To facilitate these goals, on top of the human data we provide an evaluation scheme and off-the-shelf baseline implementations including evaluations.

We provide a guide to the benchmark as well as some preliminary work on using Nocturne to design agents and test their capabilities and human-similarity. We demonstrate that learning effective agents in Nocturne is challenging; tuned RL and imitation learning baselines struggle to successfully complete the highly interactive scenes. Finally, we demonstrate that the agents achieve relatively low distance to the expert trajectories and that there does not appear to be a zero-shot coordination problem [13] at this level of agent capability.

2 Related Work

Multi-agent traffic simulation tools and benchmarks:
In terms of multi-agent driving benchmarks that require both acceleration and steering control, there are several closely related benchmarks which differ in terms of ease of implementing multi-agent interaction, being data-driven, 2D vs. 3D, or the mechanism by which they support partial observability. We summarize these differentiating features in Table 1.

The closest works to ours are BARKS [4], SMARTS [38], and MetaDrive [20]. SMARTS [38] supports multi-agent driving in a wide variety of interactive driving scenarios and offers a wide array of default human driving behaviors to use for testing autonomous vehicles. BARKS [4], like Nocturne, is a 2D goal-driven simulator with support for external datasets and contains a wide variety of large-scale scenarios and multi-agent support. Finally, MetaDrive [20] also supports real-world data and multi-agent interaction and is able to achieve 300 FPS image rendering by rendering lower fidelity images. The main differentiating feature from these works is the support for acquiring the set of objects that are visible without requiring the rendering of camera images; to our knowledge Nocturne is the only available simulator that can compute an agent's visible objects and step the



(a) A four-way stop.

(b) A straight road with merging vehicles.

(c) Unprotected turns with conflicting routes.

(d) A crowded parking lot.

Figure 2: Four scenes demonstrating the diversity of the navigable scenes in Nocturne. Colored circles represent the goal position of the corresponding colored agent. Dots represent the trajectory of the agent, with opacity increasing as time goes on. Links to videos of experts negotiating these scenes can be found at nathanlct.com/research/nocturne.

agents dynamics at above 2000+ steps-per-second. Additional simulators are summarized in Table 1.

Partially observed multi-agent benchmarks:
There are a wide variety of partially-observable multi-agent benchmarks not focused on driving that differ from our work along the axes of underlying game structure, level of partial observability, and accessibility of expert data. Within the card-playing domain, Hanabi [2] is frequently used to investigate coordination under partial observability, is fully coopeartive, and features between 2-5 agents. The Starcraft multi-agent benchmark (SMAC) [30], perhaps the most ubiquitous MARL benchmark, features many agents and high-dimensional observations but does not come with human data, is mostly fully-observed, and many algorithms now achieve perfect performance in this challenge [37]. Melting Pot [18] features a huge diversity of many-agent mixed-motive challenges but does not have available expert data. Google Research Football [17] provides a few zero-sum multi-agent environments featuring two teams consisting of several agents.

Table 1: Comparison of representative driving simulators. State-based partial observability refers to whether the set of visible objects can be queried. Expert data refers to whether expert human data is available for the scenes. Baseline human models refers to whether pre-existing human driver models are available to drive agents in the scene.

Simulator	Multi-agent Support	2D/3D	State-Based Partial Observability	Expert Data	Baseline Human Models
CARLA [9]		3D			✓
SUMMIT [6]	✓	3D			✓
MACAD [24]	✓	3D			✓
Highway-env [19]		2D			✓
Sim4CV [21]		3D			
Duckietown [26]		3D			
SMARTS [38]	✓	2D			✓
MADRaS [31]	✓	2D			✓
DriverGym [16]		2D		✓	✓
DeepDrive-Zero [28]	✓	2D			
MetaDrive [20]	✓	3D		✓	✓
VISTA [20]	✓	3D		✓	✓
Nocturne	✓	2D	✓	✓	

3 Benchmark construction

3.1 Defining a Nocturne Scene

In the following sections, we will refer to objects that can move (vehicles, pedestrians, cyclists) as *road objects* and anything that cannot move (lane lines, road edges, stop signs, etc.) as *road points*. *Road points* are connected together to form a polyline. We will refer to the type of road polyline that should not be crossed by vehicles as a **road edge**.

Nocturne scenes require a map consisting of polylines, a set of initial and final road object positions, and optionally a set of trajectories for the road objects. Nocturne currently acquires its scenes, goals, and expert trajectories from the Waymo Motion Dataset [10] but can be configured to support any dataset that represents its road features as points or polylines and that contains start and end position coordinates for any road objects. The Waymo Motion dataset consists of 487004 nine-second trajectory snippets discretized at a rate of 0.1 Hz with the first second intended to be used as context and the latter eight seconds to be used for prediction.

One challenge of selecting and constructing the scenarios for Nocturne is that these trajectories are collected by labeling the vehicles observed by a Waymo car as it drives. Consequently, we do not have a complete birds-eye view of the scene. Hence, there are cars that may have been in the scene that were not visible to the Waymo vehicle and therefore are not included in the dataset; for the same reason, the expert trajectories are also incomplete and may not persist throughout the entire duration of the rollout. In other words, the expert trajectories contain agents that unpredictably flicker in and out of existence. Similarly, the set of traffic lights that were visible to the Waymo vehicle may be insufficient to uniquely determine the underlying traffic light state. For this reason, this first version of the Nocturne benchmark comes with a few restrictions: we do not use traffic lights due to the incomplete state and for the posed challenges we do not include pedestrians or cyclists as replaying the expert trajectories may cause them to unpredictably appear on top of a vehicle trajectory and cause an unavoidable collision. We also filter out scenes that contain traffic lights which leaves the majority of the remaining scenes as roundabouts, unsignalized intersections, arterial roads, and parking lots. This leaves the benchmark consisting of 134453 snippets.

3.2 Partial Observability Model and Collision Handling

To support investigation into coordination under partial observability, agents in Nocturne come with a configurable view-cone as is shown in Fig. 1. An element (*road object* or *road point*) is considered observable if there is a single ray in the cone that intersects with that element that does not pass through any *road object* on its path to the element. Stop signs are always visible if they are within the agent view-cone even if they would be otherwise obscured on the assumption that they would be raised at a sufficiently high level. We select the angle of the cone to be 120 degrees (approximately the range of binocular vision) and 80 meters radius. We note that this does not perfectly mimic a human visual model which has additional complexities such as dynamic variations in the functional field of view [8], phenomena relating to interactions between visible objects such as crowding [5], and the occasionally necessary ability to pay attention to objects further than 80 meters away at high speeds.

The primary challenge in computing which road points and objects are observable is their relatively large number: a scene contains 30 vehicles and 4700 road edge points on average. We use a Bounding Volume Hierarchy (BVH) to maintain the road objects and select candidates for potentially observed vehicles. We build the BVH using approximate agglomerative clustering [12] to generate a high-quality BVH. When computing the observed objects, we first use the BVH to select the candidates that lie in the *axis-aligned bounding box* (AABB) of the conic view field. Since there are a comparatively larger number of all types of road points (order of 15000 on average), we use a 2D range tree [3] to maintain all of the road points. When computing the observed road points, we do a range search in the 2D range tree to select the candidates that lie in the AABB of the conic view field. For both vehicles and road points, once we have the candidates in the conic view-field, we perform a brute-force visibility check for the object and road points respectively by ray-casting from the viewing agent to all the candidate vertices; an object or road point is visible if there is a ray reaching any point of it that does not pass through any *road object*. We accelerate this procedure using the auto-vectorization property of compilers which gives a 500% speed-up to this step.

We use a similar procedure to the visibility checking to accelerate our collision detection of vehicle-vehicle and vehicle-road collisions. For vehicle-road collisions, we consider a vehicle to have collided if its body intersects with any line segment of the polylines that constitute the road edges. A vehicle-vehicle collision occurs if the vehicle rectangles intersect. For both vehicle-vehicle and vehicle-road collision detection, we query the road object BVH and a separate road line segment BVH (formed once at the beginning of the episode) to generate candidates that are then evaluated for collision.

3.3 Construction of the Partially Observable Stochastic Game

Definition of the state space

Nocturne supports two possible state representations: a rasterized image and a vectorized representation of that image. While we provide default state representations, we consider it fair game on the benchmark to use any other representation as long as only objects that are visible under the conditions in Sec. 3.2 are presented to the agent. Conforming to these consistent rules about visibility allows for a fair comparison between different algorithmic approaches.

For the default vectorized representation, we adopt a fully descriptive set of features (speed, angle, width, length, etc.) for the road objects and use the VectorNet [11] representation for the road points. We will refer to the vehicle whose observation is being returned as the *ego vehicle*. Note that by default all features that can be placed into relative coordinates are returned in relative coordinates to the ego vehicle (e.g. speed is relative speed, heading is relative heading, etc). The agent goal is set to be the final position, speed, and heading of the expert agent. An agent is considered to have achieved its goal if it is within 1 meter of the final position, within 1 $\frac{\text{m}}{\text{s}}$ of the final speed of the agent, and $.3$ radians of the final heading.

The features of the ego object are:

- The speed of the object.
- The distance and angle to the goal.
- Its width and length.
- The relative speed and heading to the target speed and heading.

Road object features are:

- The speed of the object.
- The angle between the velocity vectors of the object and the ego vehicle.

- Its width and length.
- The angle and distance to the road object's position relative to the ego vehicle.
- Its heading relative to the ego vehicle.
- A one-hot vector indicating whether the object is a vehicle, pedestrian, or cyclist.

As per the VectorNet representation, each road point is part of a discretized poly-line with an approximate discretization size of $0.5\,\mathrm{m}$ (though cross-walks and speed-bumps are discretized with a much larger spacing). To ensure that any vehicle is able to discern which road points are connected to each other, we include a vector pointing to the neighbor of the point in the polyline. As with the road object representation, all points are in the frame of the observing vehicle. Road point features are:

- The angle and distance to the road point's position relative to the ego vehicle.
- A 2D element representing the vector pointing from the current road-point to its neighbor in the polyline.
- A one-hot vector indicating whether the road point is a lane-center, a road line, a road-edge that can be collided with, a stop-sign, a crosswalk, a speed-bump, or is of unknown type.

Since likely control architectures require the input vector to be of a fixed size, we select a fixed-size subset of visible road points and objects if there are too many and pad with a vector of 0.0 if there are too few. We sort both the road points, objects, and stop signs by distance and return the $500, 16, 4$ closest ones respectively where all three values are configurable by user. We note that this leads to a fairly high dimensional state and intentionally do not prune it to be smaller: dealing with this high dimensional input is a meaningful part of the benchmark. However, since we expect that users will likely want to construct their own state vectors, utility functions are also provided to allow users to directly query the set of of observed road points and objects.

Action Space

Vehicles are driven by acceleration and steering commands that are passed to a bicycle model to update the vehicle state. The angle at which the human driver views the scene is controlled by the agent *head-tilt*. In the experiments used in this paper we use 6 discrete actions for acceleration, 21 discrete actions for steering, and 5 discrete actions for head tilt with the acceleration actions uniformly splitting $[-3, 2]\ \frac{m}{s^2}$, the steering actions between $[-0.7, 0.7]$ radians, and the head tilt between $[-1.6, 1.6]$ radians. All these bounds on the actions are configurable. For more details on the vehicle model, see Appendix Sec. A.

Environment Dynamics and Goals

The total length of the expert data is 9 seconds, discretized into steps of size 0.1 seconds. For the first 1 second of the episode, all vehicles obey the expert policy. This is used to construct a history of observations for each agent that can be used to initialize or warm up the policy. After this transitory period, the episode continues for a fixed length of $T = 80$ steps. Agents are provided with a target position, speed, and heading that are taken from the final position, speed, and heading of the expert trajectories. If an agent achieves their goal within an environment specified tolerance they receive a reward of T and are removed from the system. A vehicle will also be removed if it collides with any road edge or object.

In addition, there is a process for selecting the set of vehicles that are controlled in the environment. First, we only control vehicles that at some point have a speed above $0.05\,\frac{\mathrm{m}}{\mathrm{s}}$ and that are more than $0.2\,\mathrm{m}$ from their goal; in general, these are vehicles that need to move to get to their goal. From this set of vehicles, we remove all vehicles that are already at their goal. Next, a small number of vehicles that have infeasible goals are also set to be experts (see Sec. 3.4 for how we compute this). Finally, of the remaining vehicles, we randomly select up to a maximum of 20 of them and set the remainder to replay expert trajectories. On average there are 10 vehicles in a scene so this generally leaves most vehicles as controlled.

3.4 Unusual features of the Benchmark

Finally, for completeness we note a few oddities of the benchmark that we believe are critical for potential solution designers to be aware of. These challenges are mostly properties of labeling noise and the egocentric view under which the data was collected.

Filtered out pedestrians, cyclists, and traffic lights

While the dataset contains scenes with traffic lights, pedestrians, and cyclists, the first set of Nocturne environments, NocturneV1, operates solely on scenes that have been filtered to not include traffic lights. Additionally, when constructing the environment, we remove all pedestrians and cyclists from the scene. Future versions of the benchmark will consider joint learning of pedestrians, cyclists, and vehicles but for NocturneV1 we only consider vehicles as appropriate modeling of pedestrian and cyclist dynamics is left for future work.

Egocentric data collection

The Waymo dataset that forms the basis of the first version of Nocturne is collected by driving a sensor-equipped car and recording the trajectories of all visible vehicles. Thus, in the original data, vehicles may have trajectories that are shorter than the full 9 seconds and may only appear midway through the trajectory or appear and disappear throughout the trajectory when they are obscured from the view of the sensing car. Rather than suddenly teleport cars into the scene midway through an episode, we choose to only use cars that were visible to the sensing car at the beginning of the episode (at the conclusion of the 1 second of expert replay described in Sec. 3.3). This reduces the total number of vehicles that might appear in the episode but does not change the feasibility of any of the agent goals.

Infeasible goals

Roughly 3% of the vehicles in the dataset have an expert trajectory that crosses an impassible road. This is due to labeling error in the dataset where, for example, small gaps in road edges are occasionally missed that make the road edge crossable. To ensure a benchmark where all goals are achievable, we compute all trajectories where crossing a road edge was necessary to achieve the goal and set these vehicles to replay their expert trajectory rather than be controlled; this corresponds to 3% of all vehicles. Finally, 2% of vehicles in the dataset are initialized in a colliding state. This primarily occurs in parking lot scenes where a vehicle slightly overlaps with the boundary of a parking spot or a nearby vehicle. These vehicles are also removed.

For computing the percentage of vehicles that must cross a road edge to get to their goal, we use the following procedure. We take the vehicles and shrink their width by 0.1 and their length by 0.3. These vehicles are very thin and so do not accidentally collide with a road edge due to small errors in their position. The scaling of the length ensures that the vehicles are not placed in a starting position where they have collided. Using this procedure we calculate the 3% statistic.

Illogical goals

Occasionally, agents may have goals that seem unlikely; for example, agents may be asked to come to a full stop in the middle of a highway. While these goals may seem odd, they are actual states that were achieved by humans driving on the roadways. These odd goals are likely a consequence of unobserved objects that were not observed by the egocentric data collection process. For example, a stop in the middle of an arterial road is likely caused by a queue of unobserved vehicles.

3.5 Rules of the Benchmark

We outline here a few rules that we expect solvers to respect to ensure consistency between solutions.

- The size of the view cone is fixed to 120 degrees and a distance of 80 meters. This ensures consistency between the level of partial-observability each controller must handle. Users can also tilt the head of the driver by 90 degrees in either direction to acquire more information.
- Only the first 1 second of the trajectory can be used as context or to warm-start a memory-based controller; control of the vehicles must start at 1 second into the trajectory.
- A trajectory that successfully reaches the goal is only considered valid if it reaches the goal within the 8 second time-window. All the scenes are easily solved without a time-constraint by simply creeping forwards slowly.
- The environment comes with default rewards and observations but any amount of reward-shaping or observation sharing at training time as valid. However, at test time only informa-

tion that is directly observable to the agent can be used as input to the policy. For example, sharing information about other agents' goals would be valid at train time but not at test time.

- Adding additional map information to the agent state space beyond the information provided by default is valid.
- The bounds on the action space should respected: the acceleration should be bounded between $[-6, 6]\,\frac{\mathrm{m}}{\mathrm{s}^2}$, the change in heading should not be faster than 40 degrees per second, and driver head tilt should be maintained within 90 degrees in either direction. This rules out solutions that rapidly get to goal by using accelerations that are outside of possible vehicle speed bounds or excessively sharp turns.

We note that these rules may make some of the scenes unsolvable. For example, real human drivers are not randomly initialized into a scene with a maximum of 1 second of prior context; this context may be critical to safely navigating the scene under the time constraints. Additionally, our model of human perception is not an exact match for true human perception which can extend well beyond 80 meters under certain circumstances. It is possible that there may be missing context at this viewing distance that is crucial for safe navigation. We view these constraints as similar to the challenge of label noise in supervised datasets; our constraints may place an currently undetermined upper limit on the percentage of goals that can be achieved but the benchmark still constitutes a valid comparison between methods as long as the constraints are respected.

4 Experimental Results

We run several learning methods to demonstrate that these tasks do not appear to be easily solved even at billions of steps and have a train-test generalization gap. We also briefly investigate whether the policies appear to have a zero-shot coordination challenge wherein policies perform well when paired with agents from their seed but are incompatible with policies from a different training run. Finally, we examine the human-similarity of our trained agents.

We test the following methods:

- APPO [27] trained in multi-agent mode with a shared policy i.e. every agent is controlled by the same policy but in a decentralized fashion.
- Behavior Cloning [1, 32].

For the RL method, in addition to the fixed-bonus for achieving the goal, we add the following dense reward to encourage the agent to make progress towards the target position, speed, and heading:

$$r_t = 0.2 \times \left(1 - \frac{||x_t - x_g||_2}{||x_0 - x_g||_2}\right) + 0.2 \times \left(1 - \frac{||v_t - v_g||_2}{40}\right) + 0.2 \times \left(1 - \frac{f(h_t, h_g)}{2\pi}\right) \tag{1}$$

where x_t is the position at time-step t, x_g is the goal position, v_t is the speed at time-step t and v_g is the target speed, h_t and h_g are the current and target heading in world coordinates (i.e. not in a relative frame). $f(h_t, h_g)$ returns the minimum angle between h_t and h_g. Since coordinates are in a relative frame and the agent cannot observe the world frame, note that $f(h_t, h_g)$ is an observation provided to the agent. Note that there are values for the discount factor for which the optimal policy under the dense reward will hover near the goal and then only reach the goal at the terminal step instead of achieving it as soon as possible: the addition of the dense reward will affect the trajectory the agent will take. The use of this dense reward is not a necessary component of the benchmark and here is just used to generate good policies. For details on architecture, hyperparameters, and computational resources see Appendix Sec. B.

5 Analysis

5.1 Success rate of baselines

Fig. 3 shows the performance of the agents on the training set after $3e9$ steps which takes approximately two days. Each line corresponds to a policy trained on a fixed subset of the training scenes. We score our policies in two primary ways: the fraction of vehicles that achieve their goal (goal

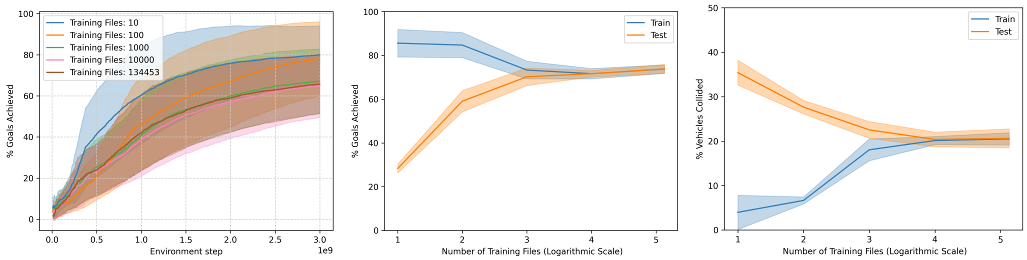


Figure 3: (Left) Success at getting to the specified goal on the training data as a function of number of environment steps. Files=X means the agent was trained on X fixed scenes sampled from the training dataset. (Middle) Percent of agents that achieved their goals. (Right) Percent of agents that collided. The logarithmic scale is base 10.

rate) and the fraction of vehicles that collide with another vehicle or road edge (collision rate). Note that this use of collision rate differs from its use in papers such as [15, 33] which compute collision rate as the fraction of scenes that contain a vehicle-vehicle collision and do not count a vehicle-road intersection as a collision.

We investigate the effect of the size of the dataset on the train and test performance. In the right half of Fig. 3 we can see that the inclusion of a larger training dataset decreases the performance of the algorithms up to 1000 files as they struggle with the diversity of the data. However, the inclusion of additional data narrows the magnitude of the train and test gap and closes it fully at 10000 files. However, it is still possible that a divergence between train and test might re-emerge as agents become more capable on the train set and approach a 100% goal rate.

Finally, Table 2 compares the APPO and BC agents. For the APPO agent, we include the results for the agent trained on 10000 training files. The expert playback row refers to replay of the expert trajectories.

Table 2: Overview of metrics across methods for an 8 second rollout.

Algorithm	Collision Rate (%)	Goal Rate (%)	ADE (m)	FDE (m)
Expert Playback	4.9	100	0	0
APPO	20.3 ± 0.8	71.7 ± 0.7	3.1 ± 0.2	6.1 ± 0.3
BC	$38.2 \pm .1$	25.3 ± 0.1	5.6 ± 0.1	9.2 ± 0.1

5.2 Human-agent trajectory similarity

We analyze the results of our experiments with respect to how human-like the resultant policies are using displacement error between expert and agent trajectories as our metric (i.e. L2 distance between the agent and expert trajectories). To align with the definition used in other works [15, 33] we disable the removal of vehicles upon collision / reaching goal. However, we note a few dissimilarities that make comparisons with other works difficult. First, agents are provided with a goal position, a feature that is often not available to other predictive methods. Second, our experts are stepped in scenes that may contain pedestrians or cyclists but our agents are replayed in the same scene without the corresponding pedestrians or cyclists. This can make the magnitude of the displacement error not directly comparable to the values in other works; the low value of the displacement error may simply indicate that a majority of the scenes have unique optima. Fig. 4 examines the average difference in position between the agent and expert trajectories averaged across 5 training runs and demonstrates that the influence of more training data on displacement error is flat after 1000 files.

5.3 Policy Failure Modes

Here we investigate the mechanisms under which our policies fail to achieve their goals and collide to shine a light on potential avenues for improvement. The key failure mode we qualitatively observe are failures in scenes in which agents are required to interact with another agent either by waiting or merging. While we cannot measure interactivity directly, we measure a proxy by looking at

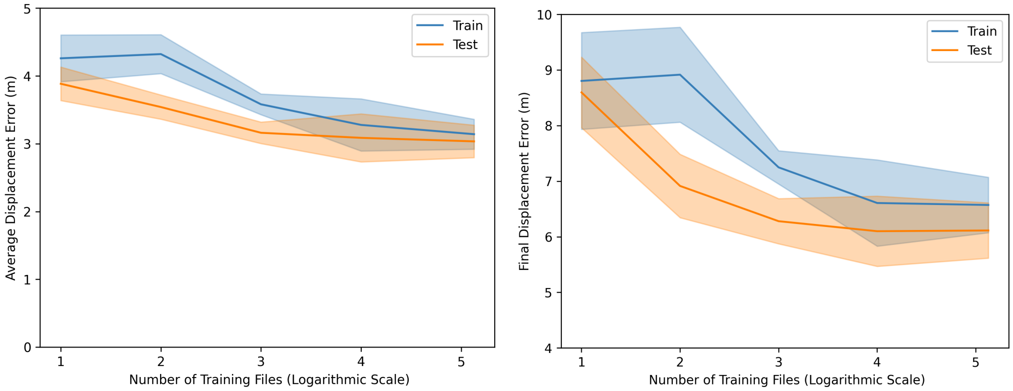


Figure 4: (Left) Average displacement error (mean l2-distance between an agent and an expert at each time-step). (Right) Final displacement error (l2-distance between an agent and an expert at the final time-step that an expert has a valid state). The logarithmic scale is base 10.

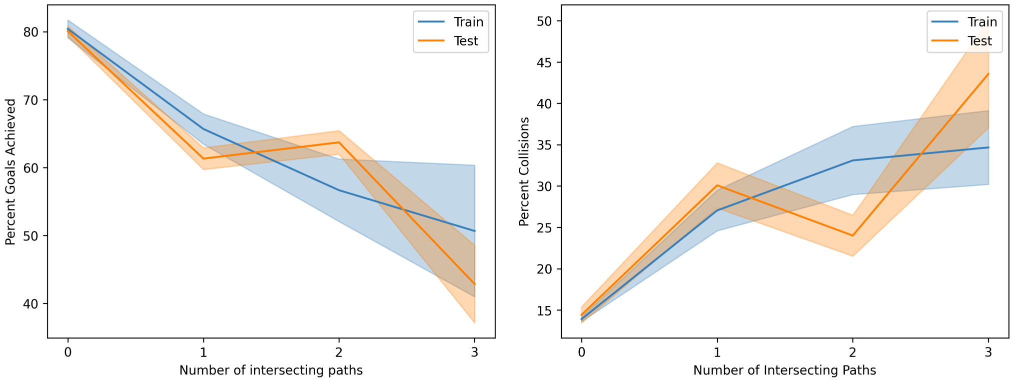


Figure 5: Goal rate (left) and collision rate (right) of vehicles as a function of the number of times that their corresponding expert trajectory intersected with another expert trajectory (intersections). As more than 3 interactions are rare, creating noisy statistics, the results are placed into the 3 interaction bin. The logarithmic scale is base 10.

the intersection of the expert trajectories. We play the experts forwards, record their trajectory as polylines, and consider a vehicle to have as many interactions as there are intersections of the vehicle's expert trajectory polyline with other vehicle's expert trajectory polylines (i.e. if two vehicles' trajectories in time cross at a point, that's an interaction). This captures interactions such as crossing at a four-way stop, merges, and others but also may unintentionally pick up interactions such as driving behind another vehicle. Close to 25% of vehicles have at least one interaction. The collision and goal rates as a function of interactions are plotted in Fig. 5 and demonstrate that the goal rate declines precipitously and the collision rate increases sharply as the number of interactions increase. This suggests that our agents have learned to get to their goals but perform poorly in settings where getting to the goal requires coordination with another agent.

5.4 Zero-shot coordination

We investigate whether the agents are learning incompatible conventions across seeds (ZSC) by taking the seeds from our best performing agent on the test set (this is the training run where we trained the agent on all files) and sampling half the agents from one seed and half from another. We then perform this procedure across all 5 seeds, running each pair on the same set of 100 files. The results of this are summarized in Fig. 6 and appear to indicate that at the current level of agent performance there does not appear to be a zero-shot coordination issue. However, it is still possible that one might emerge as the agents become more capable and learn arbitrary symmetry-breaking conventions that are not compatible across agents. This argument lines up with the results of Sec. 5.3

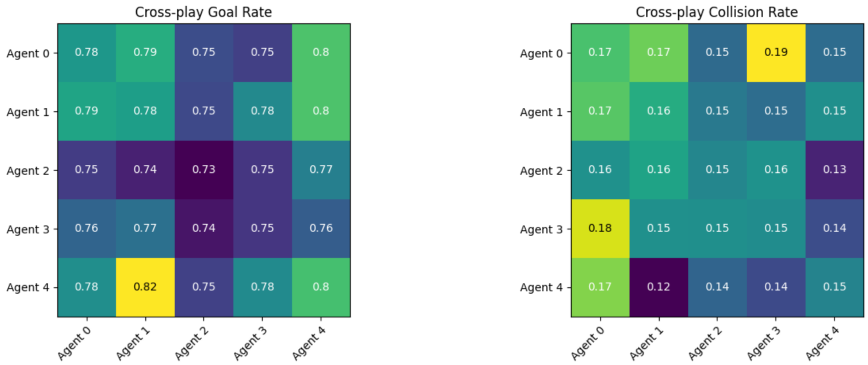


Figure 6: Goal rate (left) and collision rate (right) of agents when 50% are sampled from one side and 50% from another. The diagonal represents the baseline performance of the seed while element i, j represents seed i playing vs. seed j.

which shows that the agents fail at high rates in interactive scenes; it may be necessary to get to lower failure rates in such settings for ZSC issues to emerge.

6 Conclusion

We have introduced Nocturne, a simulator and benchmark intended to aid in the study of human-like decentralized coordination for driving systems. We present results on the applications of RL and imitation to this system, however, there still remains work to be done to build agents that operate with the collision and goal rate that humans achieve as well as how to learn these agents efficiently. Given better agents, human-like rules and conventions may be emergent properties of driving safely in these settings [23].

There is also ample remaining work to be done on new benchmarks. Due to the ego-centric data collection, we are forced to remove vehicles once they achieve their goal (i.e. the last observed position of the driver in the data). However, it may be possible to use generative models or other generative mechanisms to sample new goals for the agents to continue their trajectory once they achieve the goals set out in the data. Similarly, generative models could be used to complete the traffic light states and enable the inclusion of the traffic light scenes.

Finally, one open question is how to use Nocturne agents as predictive models of human driving. At the moment a Nocturne agent requires a goal to which it is driving. To use Nocturne agents for prediction (say for an autonomous vehicle trying to predict the motion of agents in the scene), a method for inferring goals from the 1-second context needs to be implemented. A topic for future work is to use supervised methods to predict the goals and enable fully decentralized prediction of Nocturne agents from egocentric observations.

7 Reproducibility and Ethical Statement

We do not release trained models as the Waymo Motion dataset restricts the release of trained models. While the dataset contains images that have been anonymized, only trajectory data is used in this work. The benchmark and all files needed to run it are publicly available.

8 Acknowledgements

We thank the Python community [35] for creating the core tools that enabled our work including Hydra [36], Pytorch [25], Matplotlib [14], and numpy [22, 34]. A huge thanks to Aleksander Petrenko for a ton of support in tuning and debugging SampleFactory [27]. Thanks to Scott Ettinger for help in understanding some of the peculiarities of the Waymo Motion Dataset [10]. Thanks to Rachit Singh for help with some of the results analysis scripts. We would also like to thank the International Emerging Actions project SHYSTRA (CNRS) for support of Nathan Lichtlé.

References

[1] Bain, M., and Sammut, C. A framework for behavioural cloning. In *Machine Intelligence 15* (1995), pp. 103–129.

[2] Bard, N., Foerster, J. N., Chandar, S., Burch, N., Lanctot, M., Song, H. F., Parisotto, E., Dumoulin, V., Moitra, S., Hughes, E., et al. The hanabi challenge: A new frontier for ai research. *Artificial Intelligence 280* (2020), 103216.

[3] Bentley, J. L. Decomposable searching problems. Tech. rep., CARNEGIE-MELLON UNIV PITTSBURGH PA DEPT OF COMPUTER SCIENCE, 1978.

[4] Bernhard, J., Esterle, K., Hart, P., and Kessler, T. Bark: Open behavior benchmarking in multi-agent environments. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (2020), IEEE, pp. 6201–6208.

[5] Bouma, H. Interaction effects in parafoveal letter recognition. *Nature 226*, 5241 (1970), 177–178.

[6] Cai, P., Lee, Y., Luo, Y., and Hsu, D. Summit: A simulator for urban driving in massive mixed traffic. In *2020 IEEE International Conference on Robotics and Automation (ICRA)* (2020), IEEE, pp. 4023–4029.

[7] Cho, K., Van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., and Bengio, Y. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078* (2014).

[8] Crundall, D., Underwood, G., and Chapman, P. Driving experience and the functional field of view. *Perception 28*, 9 (1999), 1075–1087.

[9] Dosovitskiy, A., Ros, G., Codevilla, F., Lopez, A., and Koltun, V. Carla: An open urban driving simulator. In *Conference on robot learning* (2017), PMLR, pp. 1–16.

[10] Ettinger, S., Cheng, S., Caine, B., Liu, C., Zhao, H., Pradhan, S., Chai, Y., Sapp, B., Qi, C. R., Zhou, Y., et al. Large scale interactive motion forecasting for autonomous driving: The waymo open motion dataset. In *Proceedings of the IEEE/CVF International Conference on Computer Vision* (2021), pp. 9710–9719.

[11] Gao, J., Sun, C., Zhao, H., Shen, Y., Anguelov, D., Li, C., and Schmid, C. Vectornet: Encoding hd maps and agent dynamics from vectorized representation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2020), pp. 11525–11533.

[12] Gu, Y., He, Y., Fatahalian, K., and Blelloch, G. Efficient bvh construction via approximate agglomerative clustering. In *Proceedings of the 5th High-Performance Graphics Conference* (2013), pp. 81–88.

[13] Hu, H., Lerer, A., Peysakhovich, A., and Foerster, J. “other-play” for zero-shot coordination. In *International Conference on Machine Learning* (2020), PMLR, pp. 4399–4410.

[14] Hunter, J. D. Matplotlib: A 2d graphics environment. *Computing in science & engineering 9*, 03 (2007), 90–95.

[15] Igl, M., Kim, D., Kuefler, A., Mougin, P., Shah, P., Shiarlis, K., Anguelov, D., Palatucci, M., White, B., and Whiteson, S. Symphony: Learning realistic and diverse agents for autonomous driving simulation. *arXiv preprint arXiv:2205.03195* (2022).

[16] Kothari, P., Perone, C., Bergamini, L., Alahi, A., and Ondruska, P. Drivergym: Democratising reinforcement learning for autonomous driving. *arXiv preprint arXiv:2111.06889* (2021).

[17] Kurach, K., Raichuk, A., Stańczyk, P., Zając, M., Bachem, O., Espeholt, L., Riquelme, C., Vincent, D., Michalski, M., Bousquet, O., et al. Google research football: A novel reinforcement learning environment. In *Proceedings of the AAAI Conference on Artificial Intelligence* (2020), vol. 34, pp. 4501–4510.

[18] Leibo, J. Z., Dueñez-Guzman, E. A., Vezhnevets, A., Agapiou, J. P., Sunehag, P., Koster, R., Matyas, J., Beattie, C., Mordatch, I., and Graepel, T. Scalable evaluation of multi-agent reinforcement learning with melting pot. In *International Conference on Machine Learning* (2021), PMLR, pp. 6187–6199.

[19] Leurent, E. An environment for autonomous driving decision-making. https://github.com/eleurent/highway-env, 2018.

[20] Li, Q., Peng, Z., Xue, Z., Zhang, Q., and Zhou, B. Metadrive: Composing diverse driving scenarios for generalizable reinforcement learning. *arXiv preprint arXiv:2109.12674* (2021).

[21] Müller, M., Casser, V., Lahoud, J., Smith, N., and Ghanem, B. Sim4cv: A photo-realistic simulator for computer vision applications. *International Journal of Computer Vision 126*, 9 (2018), 902–919.

[22] Oliphant, T. E. *A guide to NumPy*, vol. 1. Trelgol Publishing USA, 2006.

[23] Pal, A., Philion, J., Liao, Y.-H., and Fidler, S. Emergent road rules in multi-agent driving environments. *arXiv preprint arXiv:2011.10753* (2020).

[24] Palanisamy, P. Multi-agent connected autonomous driving using deep reinforcement learning. In *2020 International Joint Conference on Neural Networks (IJCNN)* (2020), IEEE, pp. 1–7.

[25] Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems 32* (2019).

[26] Paull, L., Tani, J., Ahn, H., Alonso-Mora, J., Carlone, L., Cap, M., Chen, Y. F., Choi, C., Dusek, J., Fang, Y., et al. Duckietown: an open, inexpensive and flexible platform for autonomy education and research. In *2017 IEEE International Conference on Robotics and Automation (ICRA)* (2017), IEEE, pp. 1497–1504.

[27] Petrenko, A., Huang, Z., Kumar, T., Sukhatme, G., and Koltun, V. Sample factory: Egocentric 3d control from pixels at 100000 fps with asynchronous reinforcement learning. In *International Conference on Machine Learning* (2020), PMLR, pp. 7652–7662.

[28] Quiter, C. Deepdrive zero, 2020.

[29] Rajamani, R. *Vehicle dynamics and control*. Springer Science & Business Media, 2011.

[30] Samvelyan, M., Rashid, T., De Witt, C. S., Farquhar, G., Nardelli, N., Rudner, T. G., Hung, C.-M., Torr, P. H., Foerster, J., and Whiteson, S. The starcraft multi-agent challenge. *arXiv preprint arXiv:1902.04043* (2019).

[31] Santara, A., Rudra, S., Buridi, S. A., Kaushik, M., Naik, A., Kaul, B., and Ravindran, B. Madras: Multi agent driving simulator. *Journal of Artificial Intelligence Research 70* (2021), 1517–1555.

[32] Schaal, S. Learning from demonstration. *Advances in neural information processing systems 9* (1996).

[33] Suo, S., Regalado, S., Casas, S., and Urtasun, R. Trafficsim: Learning to simulate realistic multi-agent behaviors. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2021), pp. 10400–10409.

[34] Van Der Walt, S., Colbert, S. C., and Varoquaux, G. The numpy array: a structure for efficient numerical computation. *Computing in science & engineering 13*, 2 (2011), 22–30.

[35] Van Rossum, G., and Drake Jr, F. L. *Python tutorial*, vol. 620. Centrum voor Wiskunde en Informatica Amsterdam, The Netherlands, 1995.

[36] Yadan, O. Hydra-a framework for elegantly configuring complex applications. *Github 2* (2019), 5.

[37] Yu, C., Velu, A., Vinitsky, E., Wang, Y., Bayen, A., and Wu, Y. The surprising effectiveness of ppo in cooperative, multi-agent games. *arXiv preprint arXiv:2103.01955* (2021).

[38] Zhou, M., Luo, J., Villella, J., Yang, Y., Rusu, D., Miao, J., Zhang, W., Alban, M., Fadakar, I., Chen, Z., et al. Smarts: Scalable multi-agent reinforcement learning training school for autonomous driving. *arXiv preprint arXiv:2010.09776* (2020).

A Vehicle Model

Our vehicles are driven by a kinematic bicycle model [29] which uses the center of gravity as reference point. The dynamics are as follows. Here (x_t, y_t) stands for the coordinate of the vehicle's position at time t, θ_t stands for the vehicle's heading at time t, v_t stands for the vehicle's speed at time t, a stands for the vehicle's acceleration and δ stands for the vehicle's steering angle. L is the distance from the front axle to the rear axle (in this case, just the length of the car) and l_r is the distance from the center of gravity to the rear axle. Here we assume $l_r = 0.5L$.

$$
\begin{aligned}
\dot{v} &= a \\
\bar{v} &= \text{clip}(v_t + 0.5\ \dot{v}\ \Delta t, -v_{\text{max}}, v_{\text{max}}) \\
\beta &= \tan^{-1}(\frac{l_r\ \tan(\delta)}{L}) \\
&= \tan^{-1}(0.5\ \tan(\delta)) \\
\dot{x} &= \bar{v}\ cos(\theta + \beta) \\
\dot{y} &= \bar{v}\ sin(\theta + \beta) \\
\dot{\theta} &= \frac{\bar{v}\ \cos(\beta)\ \tan(\delta)}{L} \\
x_{t+1} &= x_t + \dot{x}\ \Delta t \\
y_{t+1} &= y_t + \dot{y}\ \Delta t \\
\theta_{t+1} &= \theta_t + \dot{\theta}\ \Delta t \\
v_{t+1} &= \text{clip}(v_t + \dot{v}\ \Delta t, -v_{\text{max}}, v_{\text{max}})
\end{aligned}
$$

B Architecture, Hyperparameters, and Computational Details

The APPO experiments are each run for two days on 1 V100 GPU and 10 CPUs (Intel Xeon CPU E5-2698 v4 @ 2.20GHz) and each experiment is run for 5 seeds. For the RL experiments we run 5 conditions over 5 seeds for 2 days leading to a total of 1200 GPU hours and 12000 CPU hours. For the Imitation Learning experiments we run 5 seeds for three hours leading to 15 GPU hours and 100 CPU hours.

The architecture for the APPO agent is a two-layer neural network with 256 hidden units and Tanh non-linearities followed by a Gated Recurrent Unit [7] with 256 hidden states. The Behavior Cloning experiment is run on a single GPU for five seeds, trained on 1000 files, for 600 gradient steps with a batch size of 512. Here instead of using a recurrent neural network we stack the current state and the prior 4 states as an input and pass this through a three-layer (1025, 256, 128) neural network that then outputs two categorical heads, one for acceleration and one for steering. The acceleration head is binned into 15 bins equally spaced between $[-6, 6]\ \frac{\text{m}}{\text{s}^2}$ and steering to 43 equally spaced bins between $[-0.7, -0.7]$ radians. As there is no corresponding head tilt in the expert actions we extend the angle of the visibility cone to π radians. Note that this violates a rule on the cone shape set forth in Sec. 3.5; figuring out how to add head tilt to imitation agents remains an open question.

For APPO we use almost all the default hyperparameters of SampleFactory [27] at commit aed6cc92a7eb3510c4d4bcfac083ced07b5222f9. However, we use a batch size of 7168 and scale the observations by 10.0. For Imitation Learning we use a learning rate of $3 \cdot 10^{-4}$, a batch size of 512, and stack a total of 5 states together to endow the agents with memory.

C License Details and Accessibility

Our code is released under an MIT License. The Waymo Motion Dataset is released under a Apache License 2.0. The code is available at https://github.com/facebookresearch/nocturne.