Unlocking Efficient Vehicle Dynamics Modeling via Analytic World Models

Asen Nachkov[1], **Danda Pani Paudel**[1], **Jan-Nico Zaech**[1], **Davide Scaramuzza**[2], **Luc Van Gool**[1]

[1]INSAIT, Sofia University "St. Kliment Ohridski", Sofia, Bulgaria
[2]University of Zurich, Zurich, Switzerland

Abstract

Differentiable simulators represent an environment's dynamics as a differentiable function. Within robotics and autonomous driving, this property is used in Analytic Policy Gradients (APG), which relies on backpropagating through the dynamics to train accurate policies for diverse tasks. Here we show that differentiable simulation also has an important role in world modeling, where it can impart predictive, prescriptive, and counterfactual capabilities to an agent. Specifically, we design three novel task setups in which the differentiable dynamics are combined within an end-to-end computation graph not with a policy, but a state predictor. This allows us to learn relative odometry, optimal planners, and optimal inverse states. We collectively call these predictors Analytic World Models (AWMs) and demonstrate how differentiable simulation enables their efficient, end-to-end learning. In autonomous driving scenarios, they have broad applicability and can augment an agent's decision-making beyond reactive control.

1 Introduction

Differentiable simulation (DiffSim) has emerged as a powerful tool to train controllers and predictors across different domains like physics (Holl, Koltun, and Thuerey 2020), graphics (Laine et al. 2020), and robotics (Hu et al. 2019; Degrave et al. 2019). At its core, it is the ability to differentiate through an environment's dynamics, which in turn allows us to embed the environment within a broader computational graph. Training in such an end-to-end loop involves differentiating both through the forward passes of any modules involved, as well as the dynamics themselves.

An immediate application for this is policy learning, with the corresponding method called Analytic Policy Gradients (APG) (Nachkov, Paudel, and Van Gool 2025). For autonomous vehicle (AV) simulation, the dynamics represent the equations of motion that evolve a vehicle's state from one timestep to the next, under a specific action. To learn an optimal policy, APG repeatedly rolls out a trajectory from the current policy, and supervises it with a reference expert one. During backpropagation gradients from the difference between each realized and expert state pass through the dynamics, reach the policy and update its weights.

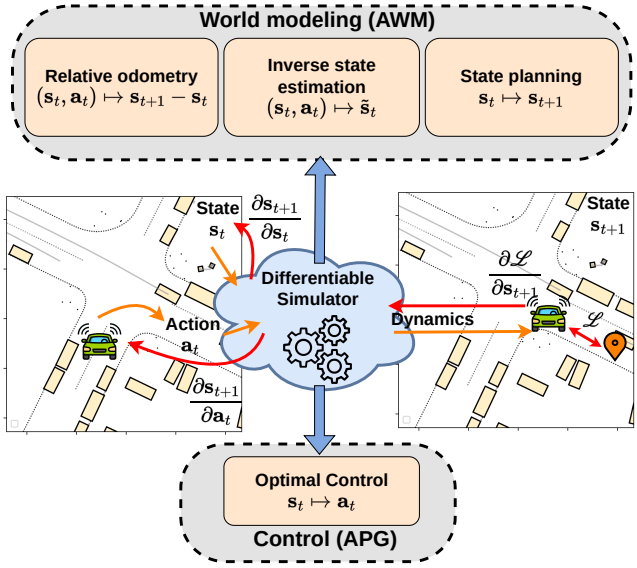

Figure 1: Differentiable simulation for world modeling. Previously, differentiable simulation has been used to train controllers using analytic policy gradients (bottom). Our contribution is in applying it for learning relative odometry, state planning, and inverse state estimation (top).

An important question is whether the applications of differentiable simulation end with policy learning. A fundamental task for any agent, especially a driving vehicle, is world modeling – predicting different states of interest like next states, desired states or counterfactual states. For a self-driving vehicle such world modeling is crucial to assess the effects of its actions as it navigates in densely populated scenarios. And similar to control, predicting the world requires understanding its dynamics, which is where world modeling connects with differentiable simulation. Thus, in this work we are interested in designing an APG-style training setup not for policy learning, but for world modeling. Our main claim is that differentiable simulation allows us to efficiently obtain accurate predictors for diverse world modeling tasks, whereas without it, one would need to rely on costly trial-and-error search.

To understand the efficiency that DiffSim provides, we consider first that in APG the gradients of the dynamics

automatically guide the policy toward the optimal actions that reproduce the expert states. There is no search involved, it is all end-to-end, unlike when the environment is treated as a black box, in which case the optimal actions have to be learned through trial-and-error (Sutton et al. 1999). Second, APG minimizes a training loss not in the action space, as methods like behavior cloning do, but in the state space. This allows any nonlinear effects in the dynamics to condition the policy, causing it to learn more physically-consistent features. We aim to capture these benefits in our world models.

There are different ways to understand the effect of one's actions. Fig. 1 shows our approach, which uses the differentiability of the simulator to formulate three task setups related to world modeling. First, the effect of an agent's action could be understood as the difference between the agent's next state and its current state. If a vehicle's state consists of its position, yaw, and velocity, then this setup has an odometric interpretation, asking the question *"Where will the agent go?"*. Second, an agent could predict a desired next state to visit, which is a form of state planning. It asks the question *"Where should it go?"*. Third, we can ask *"Given an action in a particular state, what should that state be so this action is optimal?"*, which is another form of world modeling but also an inverse problem, effectively asking the counterfactual *"Where should the agent have been?"*. All these tasks can be solved using trial-and-error approaches, as in RL, where the agent has to explore and search. Yet, our DiffSim designs solve them efficiently, without search, from direct supervision. Similar to APG, we call the corresponding predictors Analytic World Models (AWMs) to indicate they are trained using DiffSim and a state is predicted, instead of an action.

We note that some world modeling tasks like simple next-state prediction do not need a differentiable simulator to be solved efficiently. These are tasks in which the prediction targets come from the same policy as the one that was used to collect the training transitions. For any other "off-policy" tasks, DiffSim is invaluable, as we show in Sec. 3.

In our experiments we use the Waymax (Gulino et al. 2024) autonomous driving simulator, which is fully differentiable, vectorizable, functional in design, and GPU-accelerated. It is also data-driven, as it replays scenarios from the realistic, large-scale Waymo Open Motion Dataset (WOMD) (Ettinger et al. 2021). Specifically, our AWMs are trained for the freely controlled ego-vehicle, whose realized trajectories can deviate from the historical ones, while all other agents evolve according to their historical motion in the dataset. The trained agent can reason about the world dynamics with its AWMs, is lightweight, with just over 6M parameters, and runs in real-time. Training the AWMs is similarly cheap.

Our work covers diverse experimental settings. We aim to highlight the breadth of the application of DiffSim and to that end we purposefully provide separate, independent experiments for the AWM tasks. At test time the agent only has access to its learned modules. Our contributions are:

- We propose Analytic World Models (AWMs) – different state predictors trained with differentiable simulation.
- We evaluate them in diverse settings, improving over relevant baselines and previous works.

2 Related Work

Our work uses differentiable simulation for world modeling, within an AV setting. We cover the relevant context below.

Differentiable simulation. Differentiable simulators have grown in popularity because they allow one to solve ill-posed inverse problems related to the dynamics. An object's physical parameters like mass, friction, and elasticity could be estimated directly from videos and real-world experiments (de Avila Belbute-Peres et al. 2018; Murthy et al. 2020), or simulations of soft material cutting could enable precise calibration and policy learning (Heiden et al. 2021). Simulations can be parallelized across accelerators to enable efficient scaling of problem and experiment sizes (Xu et al. 2022; Macklin 2022; Freeman et al. 2021). Within the field of robotics, differentiable simulation is used extensively, especially for training robotic policies in physically-realistic settings (Newbury et al. 2024; Lutter et al. 2021). The focus has often been on object manipulation (Li et al. 2023b; Xu et al. 2021) which requires having differentiable contact models for detecting collisions. The Analytic policy gradients (APG) method has been used to train policies for trajectory tracking and navigation in quadrotors and fixed-wing drones (Wiedemann et al. 2023), quadruped locomotion (Song, Kim, and Scaramuzza 2024), and for quadrotor control from visual features (Heeg, Song, and Scaramuzza 2024).

Autonomous vehicles. For AVs simulators are crucial. Photorealistic simulators (Dosovitskiy et al. 2017; Martinez et al. 2017; Li et al. 2023a) have been used for visual control (Codevilla et al. 2018; Zhang et al. 2019), with less focus on differentiable motion ones (Lavington et al. 2024). Some are differentiable but lack expert actions (Lavington et al. 2024), others are lacking acceleration support, crucial for large scale training (Sun et al. 2022; Li et al. 2022; Vinitsky et al. 2022). Waymax (Gulino et al. 2024) is a differentiable data-driven simulator for vehicle motion. It allows one to simulate trajectories instantiated from real driving scenarios and compare them to the historical human drivers' motion, which are considered ground-truth. It also provides inverse kinematics – computing the action that transfers the simulator to a particular next state, which is valuable for our AWMs.

Baseline. The work most relevant to ours is Analytic Policy Gradients (APG) (Nachkov, Paudel, and Van Gool 2025). It trains policies in a supervised manner, relying on the differentiability of the Waymax simulator. The model selects actions autoregressively from the observed agent locations, roadgraph points, traffic lights, and goal heading. At training time it learns to select those actions that would bring the simulated trajectory as close as possible to the expert trajectory. Due to its RNN architecture, the derivatives of the dynamics from each timestep mix with those of the RNN hidden state and propagate backwards to the start of the trajectory. Our goal is to generalize this approach to world modeling.

Model-based methods. Action-selection using world models (Schrittwieser et al. 2020; Ha and Schmidhuber 2018; Moerland et al. 2023) is a common problem with two main approaches: model-predictive control (MPC) (Bertsekas 2012) and Dyna-style imagination (Sutton 1991). With MPC (Arroyo et al. 2022; Romero, Song, and Scaramuzza 2024), one starts with a random policy from which actions are sampled

and evaluated. Then, the policy is repeatedly refit on only the best trajectories, from which new trajectories are sampled. Eventually, an aggregated action from the best trajectories is selected and executed. Being closed-loop, this strategy is repeated at every timestep. To assess the possibility of using our world models for action selection beyond reactive settings, in Sec. 4.2 we perform an experiment where we adopt MPC as the main action selection framework, while the learned AWMs are used to predict and score the trajectories.

3 Method

Notation. We represent the current simulator state with $\mathbf{s}_t$, the current action with $\mathbf{a}_t$, the log (= human expert = ground-truth = reference) state with $\hat{\mathbf{s}}_t$, the log action with $\hat{\mathbf{a}}_t$. The simulator is a function $\text{Sim} : \mathcal{S} \times \mathcal{A} \to \mathcal{S}$, with $\text{Sim}(\mathbf{s}_t, \mathbf{a}_t) = \mathbf{s}_{t+1}$, where the set of all states is $\mathcal{S}$ and that of the actions $\mathcal{A}$. The mapping $\text{InvKin}(\mathbf{s}_t, \mathbf{s}_{t+1}) = \mathbf{a}_t$ is called inverse kinematics and produces the action that transfers one state into another. We denote policies as π_θ and world models as f_ϕ, with θ and ϕ representing their parameters.

Strategy. To understand how DiffSim unlocks the efficient solving of diverse world modeling tasks we will show it is difficult to obtain similar predictors without DiffSim. Training there needs to happen by collecting transitions to train on, and obtaining the labels to supervise with. The main questions are: (i) how will the action selection during training happen, i.e. how to perform the rollout, and (ii) what variables do we supervise with. We will show that without DiffSim, key variables cannot be obtained, and hence one would need to rely on trial-and-error training, which is sample-inefficient.

3.1 Preliminaries – APG

It is shown in (Nachkov, Paudel, and Van Gool 2025) that a differentiable simulator can turn the unsupervised search problem of optimal policy learning into a supervised one. Here the policy π_θ produces an action $\mathbf{a}_t$ from the current state, which is executed in the environment to obtain the next state $\mathbf{s}_{t+1}$. Comparing it to the reference trajectory $\hat{\mathbf{s}}_{t+1}$ produces a loss, whose gradient is backpropagated through the simulator and back to the policy:

$$\min_\theta \left\| \text{Sim}\big(\mathbf{s}_t, \pi_\theta(\mathbf{s}_t)\big) - \hat{\mathbf{s}}_{t+1} \right\|_2^2. \tag{1}$$

The key gradient here is that of the next state with respect to the current agent actions $\frac{\partial \mathbf{s}_{t+1}}{\partial \mathbf{a}_t}$. The loss is minimized whenever the policy outputs an action equal to the inverse kinematics $\text{InvKin}(\mathbf{s}_t, \hat{\mathbf{s}}_{t+1})$, which is also what the policy implicitly learns. To obtain similar supervision without DiffSim, one would need to supervise the policy with the inverse kinematic actions, which are unavailable if the environment is considered a black box. Hence, this is an inverse problem that is not efficiently solvable without access to a known environment, in this case to provide inverse kinematics.

Furthermore, DiffSim is beneficial in APG because the loss in Eqn. 1 is applied in the state space, as opposed to the action space. Fig. 2 shows schematically the effects of this.

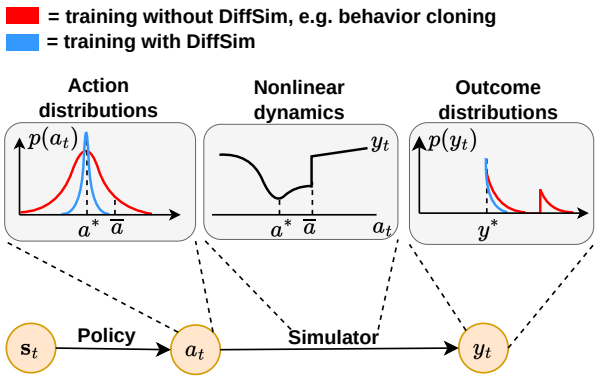

Figure 2: The benefits of differentiable simulation. Methods that do not use DiffSim, e.g. behavior cloning, shown in red, are trained to minimize a loss in the action space. If the dynamics are nonlinear (here with a jump at the action $\bar{a}$), the distribution of the outcome could be bad. DiffSim-based methods (blue) minimize a loss directly in the outcome space and the learned action distributions are tighter.

3.2 Relative Odometry

In this setting a world model $f_\phi^O : \mathcal{S} \times \mathcal{A} \to \mathcal{S}$ predicts the next state $\mathbf{s}_{t+1}$ from the current state-action pair $(\mathbf{s}_t, \mathbf{a}_t)$. A differentiable simulator is not strictly needed to learn such a predictor. One can obtain $(\mathbf{s}_t, \mathbf{a}_t, \mathbf{s}_{t+1})$ tuples by rolling out a policy and then supervising the predictions with the next state $\mathbf{s}_{t+1}$. However, this would not utilize the available dynamics in any way. Hence we provide a formulation for bringing the simulator into an end-to-end training loop:

$$\min_\phi \left\| \text{Sim}^{-1}\big(f_\phi^O(\mathbf{s}_t, \mathbf{a}_t), \mathbf{a}_t\big) - \mathbf{s}_t \right\|_2^2. \tag{2}$$

Here, the world model f_ϕ^O takes $(\mathbf{s}_t, \mathbf{a}_t)$ and returns a next-state estimate $\tilde{\mathbf{s}}_{t+1}$. We then feed it into an inverse simulator Sim^{-1} which is a function with the property that $\text{Sim}^{-1}(\text{Sim}(\mathbf{s}_t, \mathbf{a}_t), \mathbf{a}_t) = \mathbf{s}_t$. This output is compared with the current $\mathbf{s}_t$. The loss is minimized when f_ϕ^O predicts exactly $\mathbf{s}_{t+1}$, thus becoming a predictor of the next state, conditional on the provided action $\mathbf{a}_t$.

We implement the inverse simulator for the bicycle dynamics in Waymax. They embody the most realistic nonlinear vehicle motion available in the simulator. The velocities v_x and v_y are tied to the yaw angle α of the agent through the relationship $v_x = v \cos\alpha$ and $v_y = v \sin\alpha$, where v is the current speed. However, at the first simulation step, due to WOMD being collected with noisy estimates of the agent state parameters, the relationships between v_x, v_y, and α do not hold. Thus, the inverse simulator produces incorrect results for the first timestep.

For this reason, we change the design used in the experiments to one that only requires access to a forward simulator:

$$\min_\phi \left\| \text{Sim}\big(\mathbf{s}_{t+1} - f_\phi^O(\mathbf{s}_t, \mathbf{a}_t), \mathbf{a}_t\big) - \mathbf{s}_{t+1} \right\|_2^2, \tag{3}$$

where f_ϕ^O predicts the relative state difference that executing $\mathbf{a}_t$ will bring to the agent. One can verify that the loss is

minimized if the prediction is equal to $\mathbf{s}_{t+1} - \mathbf{s}_t$. This can still be interpreted as a world model where f_ϕ^O learns to estimate how an action would change its relative state. Since the time-varying elements of the agent state consist of (x, y, v_x, v_y, α), this world model has a clear relative odometric interpretation. Learning such a predictor without a differentiable simulator will prevent the gradients of the environment dynamics from mixing with those of the network, which is suboptimal.

Inverse dynamics and inverse kinematics. Given tuples $(\mathbf{s}_t, \mathbf{a}_t, \mathbf{s}_{t+1})$, one can learn inverse dynamics $(\mathbf{s}_{t+1}, \mathbf{a}_t) \mapsto \mathbf{s}_t$ and inverse kinematics $(\mathbf{s}_t, \mathbf{s}_{t+1}) \mapsto \mathbf{a}_t$ without a differentiable simulator (Pathak et al. 2017), but this is still completely agnostic in terms of how the data was generated. Formulations that involve the simulator are also possible. We do not list them here because they are similar to Eqn. 3.

3.3 Optimal Planners

We call the mapping $f_\phi^P : \mathcal{S} \to \mathcal{S}$ with $\mathbf{s}_t \mapsto \mathbf{s}_{t+1} - \mathbf{s}_t$ a planner because it plans out the next state to visit from the current one. Unlike a policy, which selects an action without explicitly knowing the next state, the planner does not execute any actions. Until an action is executed, its output is inconsequential. We consider the problem of learning an optimal planner with respect to the expert trajectories. With a differentiable simulator we can formulate the problem as:

$$\min_\phi \left\| \mathrm{Sim}\Big(\mathbf{s}_t, \mathrm{InvKin}\big(\mathbf{s}_t, \mathbf{s}_t + f_\phi^P(\mathbf{s}_t)\big)\Big) - \hat{\mathbf{s}}_{t+1} \right\|_2^2. \quad (4)$$

Here, f_ϕ^P predicts the next state to visit as an offset to the current one. The action that reaches it is obtained using the inverse kinematics. After executing that action we directly supervise with the optimal next state. The gradient of the loss goes through the simulator, the inverse kinematics, and finally through the state planner network. Note that with a black box environment we can still supervise the planner directly with $\hat{\mathbf{s}}_{t+1}$, but a black box does not provide any inverse kinematics, hence there is no way to perform trajectory rollouts, unless with a separate behavioral policy.

3.4 Inverse Optimal State Estimation

We now consider the following task *"Given $(\mathbf{s}_t, \mathbf{a}_t)$, find an alternative state $\tilde{\mathbf{s}}_t$ for the current timestep t where taking action $\mathbf{a}_t$ will lead to an optimal next state $\hat{\mathbf{s}}_{t+1}$"*. This represents the counterfactual statement "Had the agent been in $\tilde{\mathbf{s}}_t$, then the action $\mathbf{a}_t$ would have been optimal". We formulate the learning objective as

$$\min_\phi \left\| \mathrm{Sim}\big(\mathbf{s}_t + f_\phi^I(\mathbf{s}_t, \mathbf{a}_t), \mathbf{a}_t\big) - \hat{\mathbf{s}}_{t+1} \right\|_2^2, \quad (5)$$

where f_ϕ^I needs to estimate the effect of the action $\mathbf{a}_t$ and predict a new state $\tilde{\mathbf{s}}_t$, relatively to the current state $\mathbf{s}_t$, such that after executing $\mathbf{a}_t$ in it, the agent reaches $\hat{\mathbf{s}}_{t+1}$. The loss is minimized if f_ϕ^I predicts $\tilde{\mathbf{s}}_t - \mathbf{s}_t$. The key gradient, as in Eqns. 2, 3, and 4, is that of the next state with respect to the current state. Given the design of the Waymax simulator, these gradients are readily-available.

Consider solving this task with a black box environment. To do so, one would need to supervise the prediction $f_\phi^I(\mathbf{s}_t, \mathbf{a}_t)$ with some particular state $\tilde{\mathbf{s}}_t - \mathbf{s}_t$, with $\tilde{\mathbf{s}}_t$ being unknown. By definition $\tilde{\mathbf{s}}_t = \mathrm{Sim}^{-1}(\hat{\mathbf{s}}_{t+1}, \mathbf{a}_t)$, which is unobtainable since under a black box environment assumption, Sim^{-1} is unavailable. Hence, this is another inverse problem which is not efficiently solvable unless we are given more information about the environment's inverse function.

Type	Predict	Action in Rollout	Supervise with
APG	Opt. action $\mathbf{a}_t$	From policy	inv. kin.
AWM	Next state $\mathbf{s}_{t+1}$	From policy	$\mathbf{s}_{t+1}$
	Opt. next state	From inv. kin.	$\hat{\mathbf{s}}_{t+1}$
	Inv. opt. state	From policy	Sim^{-1}

Table 1: Characteristics of learning inverse dynamics predictors without DiffSim. A black box environment is missing the computations highlighted in red and training without DiffSim would require sample-inefficient search.

The utility of this task is in providing a *confidence* measure to an action. If the prediction of f_ϕ^I is close to $\mathbf{0}$, then the agent is relatively certain that the action $\mathbf{a}_t$ is close to optimal. Likewise, a large prediction from f_ϕ^I indicates a belief that the state $\mathbf{s}_{t+1}$ after $(\mathbf{s}_t, \mathbf{a}_t)$ will be far from the expert one.

Summary of DiffSim strengths. Table 1 summarizes how training such predictors would work without DiffSim. It highlights that with a black box environment it is not possible to explicitly obtain the target variables with which to supervise the predictors. Therefore, one has to rely on exploration and trial-and-error search. Conversely, the white-box nature of a differentiable simulator allows us to (i) sidestep the trial-and-error learning, unlocking efficient training, and (ii) incorporate the dynamics within an end-to-end training loop, which helps predictors learn more physically-consistent features.

4 Experiments

To cover the range of applications of DiffSim, we provide a comprehensive suite of experiments. We evaluate the four differentiable simulation tasks separately in Sec. 4.1. In Sec. 4.2 we show joint evaluation based on model-predictive control.

AWM agent architecture. The inputs to the driving agent include the locations of all other traffic participants, the nearest roadgraph points, the traffic lights, the ego vehicle's own velocity, and any route features (heading angle towards destination, or final (x, y) waypoint) to mark the goal destination. A scene encoder extracts features from all these modalities and fuses them. A recurrent component evolves these state features in time. The three AWMs and the policy are implemented as four parallel heads on top of these features. They do not share parameters (here indicated generically as θ and ϕ). The policy is trained using Analytic Policy Gradients (APG) and its collected data is used to train the AWMs. The loss functions are Eqns. 1, 3, 4, and 5.

Metrics. To evaluate the quality of realized trajectories, we compute the average (over timesteps) displacement error (ADE) of a realized trajectory compared to the expert one. A key aspect throughout the experiments is route condition-

Model	ADE ↓	overlap ↓	offroad ↓
DQN	9.8300	0.0650	0.0370
BC	3.6000	0.1120	0.1360
Wayformer	2.3800	0.1070	0.0790
APG (previous)	2.0083	0.0800	0.0282
APG (ours)	**1.8121**	**0.0669**	**0.0263**

Table 2: APG performance. The agent is conditioned on the heading towards the final destination. *Takeaway*: our APG implementation outperforms RL, behavior cloning, sequence prediction methods, as well as the previous APG baseline.

Rollouts	min ADE ↓	min overlap ↓	min offroad ↓
1	3.5725	0.2229	0.1224
4	2.0225	0.1350	0.1130
16	1.3361	0.0956	0.1056
32	**1.1414**	**0.0840**	**0.1030**

Table 3: Retaining the multimodality of the policy. *Takeaway*: with no route conditioning, the action distribution is wide. The more trajectories we sample, the closer is one of them to the historic one, which means the agent successfully learns action distributions similar to the human expert's.

APG (ours)	Traffic BotsV1.5	MVTE	CogniBOT v1.5	GUMP	Behavior GPT
1.141	1.883	1.677	1.883	1.604	1.415

Table 4: MinADE comparison with multi-agent methods. All methods report the best of 32 modes. Ours is evaluated only on the ego-agent. *Takeaway*: our APG implementation is competitive to SOTA methods, though differences in the experimental setup limit perfect comparison.

ing. In some experiments we purposefully do not condition the agent on any form of navigation and the action distributions from the policy are expected to be wide, to cover multiple reasonable trajectories. In those cases we realize multiple trajectories and report the minimum ADE among them (min ADE). This measures whether the learned policy can cover any reasonable expert trajectory (Montali et al. 2024). Whenever route conditioning is used and the agent knows the intended destination, we only realize a single trajectory and report its ADE. We also report the minimum overlap and offroad rates. They equal the proportion of scenarios in which at least one collision or offroad event occurs within the trajectory of lowest ADE.

4.1 Evaluating the Analytic Predictors

Optimal control. For optimal control, we evaluate the policy head of our driving agent. Table 2 shows that our trajectories obtained from rolling out a policy trained with APG are accurate. Compared to the previous APG baseline (Nachkov, Paudel, and Van Gool 2025), our training procedure improves ADE by 9%. Here the agent knows the heading towards the final destination and all methods are directly comparable. The standard deviation of the ADE across 5 random seeds is 3.7×10^{-4}, hence variability plays almost no role.

Our second optimal control experiment validates not whether the agent can reach a specific destination, but whether it can drive similar to a human without being provided with an intended destination (i.e. no route conditioning). Here we realize multiple trajectories from the stochastic policy and report the best one. To improve performance we have modified the baseline APG, which we now describe.

The baseline APG suffers from a Gaussian collapse because the policy, parametrized as a *Gaussian mixture with 6 components*, samples a particular action *from across all of them*. This causes the individual Gaussians to eventually stack on top of each other. Inspired by (Nayakanti et al. 2023) instead, we sample the action from only that Gaussian, whose mean will bring the agent closest to the next expert state. During training the gradients in the backward pass only reach this component (winner-take-all), which completely prevents the Gaussian mixture from losing its multimodality. Table 3 shows that as the number of trajectories rolled out increases, the best one improves in performance. This is evidence that the policy covers the expert trajectories well.

In this setting with no route conditioning, we can compare also to state-of-the-art methods from the Waymo Sim Agents challenge (Montali et al. 2024), in Table 4. They predict on all traffic participants – not only the ego-vehicle – which is different from our single-agent setup. Nonetheless, we find the comparison useful and our method very competitive. TrafficBots (Zhang, Sakaridis, and Van Gool 2024), MVTE (Wang, Zhao, and Yi 2023), CogniBOT, GUMP, and BehaviorGPT (Zhou et al. 2024) constitute strong transformer models focused on architectural innovations. Compared to them, our method has a simple recurrent architecture but is trained using the differentiable dynamics. We leave the adaptation of AWMs to multi-agent settings as future work.

Relative odometry. The odometry head is used to imagine the locations obtained from a sequence of ego-actions. Thus, to evaluate it, we measure how similar is the imagined future to the real one. We first produce qualitative results that demonstrate the odometric world model's controllability. To do so, we condition the agent's behavior by fixing its actions (instead of sampling them from the policy) to intentionally commit to a turn over a long time frame. Concurrently, the *odometry predictor is used to autoregressively imagine* the next second (10 timesteps) of the planned motion, conditional on these fixed actions. We judge the imagined trajectory to be accurate if the imagination precisely aligns with the realized trajectory. Fig. 3 shows examples. If we condition the agent to turn left/right, accelerate/decelerate, the executed trajectory follows this motion, but importantly, the imagined trajectories for these action commands also represent similar motion. This means that the agent can accurately imagine the motion resulting from an action sequence.

Manually fixing the action sequence could easily lead to out-of-distribution state-action trajectories, as in Fig 3. For example, driving offroad, making sudden sharp U-turns, or

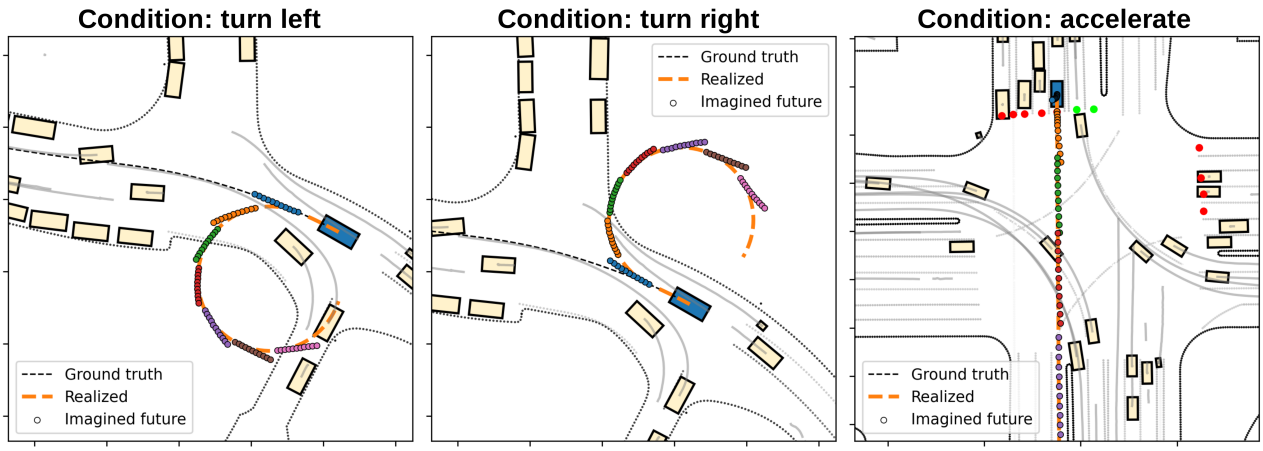

Figure 3: Predictions from the relative odometry. We condition the ego-agent (blue) to go offroad, turn, or accelerate. The imagined trajectories, shown as scattered colored circles, represent the imagined future locations of the ego-vehicle in the next 1 second, plotted in different colors at the times 1s, 2s, ..., 7s throughout the episode. They align with the actual realized trajectory, which implies that the agent can imagine its future motion accurately. The ground truth historic trajectory is added for reference.

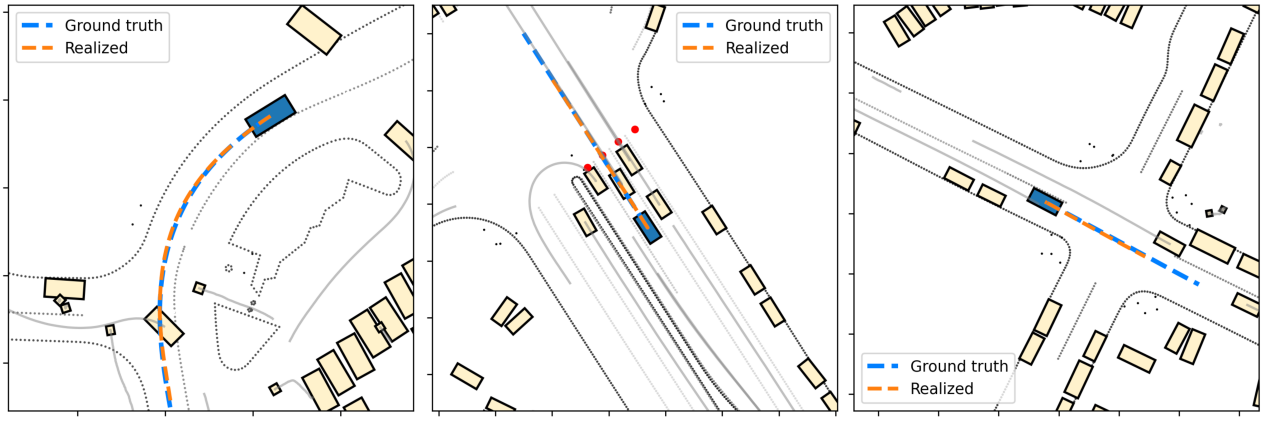

Figure 4: Trajectories obtained by using the optimal planner. They are realistic and resemble those from the policy. Training such planners is possible due to the analytically available dynamics and inverse kinematics, to drive the action selection.

maximally accelerating can be considered rare events within the expert distribution. The accurate alignment between the imagined trajectory and the executed one shows that the network learns to generalize effectively. For in-distribution sequences where actions come from the policy, and for shorter future horizons, the odometry is accurate (see Table 5).

Optimal planners. The optimal planner has a prescriptive role in that it predicts desired next states and the agent relies on the simulator's inverse kinematics to find the action that reaches them. We evaluate the planner with a combination of quantitative and qualitative metrics. Table 6 shows that obtaining an optimal planner using a differentiable environment results in strong performance, improving over the baseline APG method on all metrics. For simplicity, the planner is deterministic and actions are chosen without any sampling. When training begins, the resulting actions, obtained using the inverse kinematics, have large magnitude. The default inverse kinematics in Waymax clip them, which prevents gradients from flowing back. Thus, we disable the action clipping in the inverse kinematics only when training.

Fig. 4 shows qualitative trajectories. They are smooth and realistic. When we condition the agent on the heading angle to the target, but not the distance to it, errors are mostly longitudinal and occur from over- and under-accelerating. Some turns are sharper but still follow reasonable trajectories.

Inverse optimal state prediction. Finding a state in which a given action is optimal, is an inverse task. To motivate the setup for its evaluation, consider that if we start from an expert state $\hat{\mathbf{s}}_t$ and the selected action is optimal, $\mathbf{a}_t = \hat{\mathbf{a}}_t$, then the state we seek has a displacement of $\mathbf{0}$ from the given state $\hat{\mathbf{s}}_t$. Thus, if we assume the given actions are similar to the expert ones, the predicted displacement will indicate how far the ego-vehicle is from the current expert state $\hat{\mathbf{s}}_t$. The key quantity we look at is the norm of the predicted displacement $\|f^I_\phi(\mathbf{s}_t, \mathbf{a}_t)\|_2$. First we provide qualitative evaluation of this norm in Fig. 5. The results are meaningful – as the

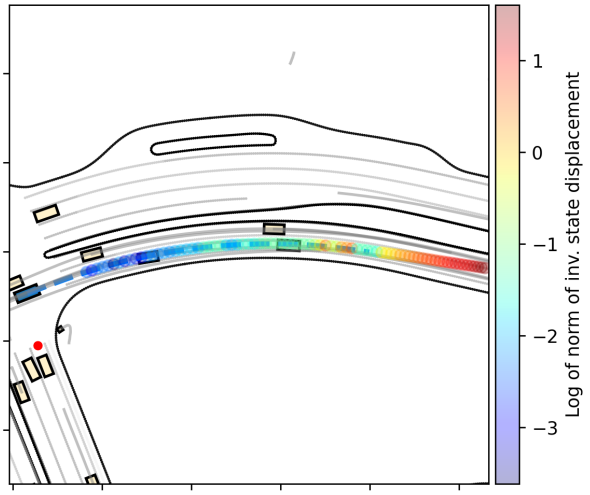

Figure 5: Realized trajectory colored according to the log-norm of the predicted inverse state displacements. Since the ego-vehicle drives faster than the expert, the norm of the optimal inverse state predictions gradually increases.

agent over-accelerates, the optimal historical trajectory starts lagging behind the realized one. The predicted inverse state displacement relative to the current state also increases.

The norm of the predicted displacement can also be used as a confidence-based metric from which to select actions. In Table 7 we evaluate a model-predictive control setting (described next) where the agent selects those actions to execute that have the lowest predicted inverse state norm. Results show this setup is as accurate as when using the distance to the next log-state, which validates that the inverse state prediction can be used to select the right actions.

4.2 Model Predictive Control (MPC)

In the real world the agent can execute only one trajectory. Yet, the world models allow it to imagine multiple trajectories and select a single action refined from them. We use model-predictive control (MPC) as an experiment in this direction. At test time the agent uses the learned world modeling predictors to autoregressively imagine a number of future trajectories and to score them (according to their inverse state norms). The action to execute is obtained by aggregating the first actions from the k-best trajectories. Importantly, this approach uses world modeling at test time to motivate the action selection. It goes beyond simple reactive decision-making because now the agent has to imagine the action's outcome and search for those actions with the best outcomes.

In Table 8 we condition on the expert heading. Increasing the number of imagined trajectories (rollouts) and their lengths (future steps) improves performance compared to the reactive single-rollout case. Imagining 8 possible futures for the next 1 second (at 10 timesteps per second) improves over the reactive setting by about 10%. Thus, the world models allow the agent to play out multiple imagined trajectories and refine its actions from them, potentially leading to safer model-based action selection.

Future steps	With DiffSim	Without DiffSim
5 (0.5 sec)	0.1698	0.3100
10 (1 sec)	0.3475	0.7900
15 (1.5 sec)	0.5496	1.6200

Table 5: ADE [m] vs time horizon. We measure the in-distribution (actions predicted by a trained policy) average distance between the imagined trajectory and the realized trajectory. *Takeaway*: the accuracy of the imagination scales better with DiffSim, compared to without.

Setting	ADE ↓	overlap ↓	offroad ↓
APG (previous)	2.0083	0.0800	0.0282
Planner, Sec. 3.3	**1.8734**	**0.0719**	**0.0254**

Table 6: Reactive evaluation of the planner. Our design outperforms the previous APG method. *Takeaway*: our planner, enabled by the corresponding AWM design, outperforms the previous APG baseline.

Rewards	ADE ↓	overlap ↓	offroad ↓
Negative distance to next log state	**1.8136**	**0.0645**	0.0226
Positive distance to next log state	1.8247	0.0649	0.0229
Negative norm of inverse state	1.8138	0.0647	**0.0218**

Table 7: Using the inverse state predictions. *Takeaway*: predictions from the inverse state estimation (last row) can be as useful as the explicit rewards for action selection.

Rollouts, (top-k)	**Future steps**	ADE ↓	overlap ↓	offroad ↓
1 (1)	1	3.5883	0.1842	0.0398
4 (1)	10	3.5975	0.1661	0.0348
8 (3)	10	3.4719	0.1760	0.0369
8 (3)	20	**3.2179**	0.1576	0.0329
12 (4)	10	3.5258	0.1783	0.0369
12 (4)	20	3.2486	**0.1550**	**0.0318**

Table 8: Model-predictive control at test time. *Takeaway*: increasing the imagined rollouts improves results.

5 Conclusion

We formulated three tasks that use differentiable simulation for world modeling. The relative odometry is *predictive*. The optimal planner is *prescriptive* (outputs a desired next state). And The inverse optimal state allows for counterfactual estimation acting as a confidence metric for the agent's own actions. We evaluated the corresponding AWMs in diverse settings and have shown that differentiable simulation unlocks the efficient learning of diverse world modeling predictors.

Acknowledgements

This research was partially funded by the Ministry of Education and Science of Bulgaria (support for INSAIT, part of the Bulgarian National Roadmap for Research Infrastructure).

References

Arroyo, J.; Manna, C.; Spiessens, F.; and Helsen, L. 2022. Reinforced model predictive control (RL-MPC) for building energy management. *Applied Energy*, 309: 118346.

Bertsekas, D. 2012. *Dynamic programming and optimal control: Volume I*, volume 4. Athena scientific.

Codevilla, F.; Müller, M.; López, A.; Koltun, V.; and Dosovitskiy, A. 2018. End-to-end driving via conditional imitation learning. In *2018 IEEE international conference on robotics and automation (ICRA)*, 4693–4700. IEEE.

de Avila Belbute-Peres, F.; Smith, K.; Allen, K.; Tenenbaum, J.; and Kolter, J. Z. 2018. End-to-end differentiable physics for learning and control. *Advances in neural information processing systems*, 31.

Degrave, J.; Hermans, M.; Dambre, J.; and Wyffels, F. 2019. A differentiable physics engine for deep learning in robotics. *Frontiers in neurorobotics*, 13: 6.

Dosovitskiy, A.; Ros, G.; Codevilla, F.; Lopez, A.; and Koltun, V. 2017. CARLA: An open urban driving simulator. In *Conference on robot learning*, 1–16. PMLR.

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

Freeman, C. D.; Frey, E.; Raichuk, A.; Girgin, S.; Mordatch, I.; and Bachem, O. 2021. Brax–a differentiable physics engine for large scale rigid body simulation. *arXiv preprint arXiv:2106.13281*.

Gulino, C.; Fu, J.; Luo, W.; Tucker, G.; Bronstein, E.; Lu, Y.; Harb, J.; Pan, X.; Wang, Y.; Chen, X.; et al. 2024. Waymax: An accelerated, data-driven simulator for large-scale autonomous driving research. *Advances in Neural Information Processing Systems*, 36.

Ha, D.; and Schmidhuber, J. 2018. World models. *arXiv preprint arXiv:1803.10122*.

Heeg, J.; Song, Y.; and Scaramuzza, D. 2024. Learning Quadrotor Control From Visual Features Using Differentiable Simulation. *arXiv preprint arXiv:2410.15979*.

Heiden, E.; Macklin, M.; Narang, Y.; Fox, D.; Garg, A.; and Ramos, F. 2021. Disect: A differentiable simulation engine for autonomous robotic cutting. *arXiv preprint arXiv:2105.12244*.

Holl, P.; Koltun, V.; and Thuerey, N. 2020. Learning to control pdes with differentiable physics. *arXiv preprint arXiv:2001.07457*.

Hu, Y.; Anderson, L.; Li, T.-M.; Sun, Q.; Carr, N.; Ragan-Kelley, J.; and Durand, F. 2019. Difftaichi: Differentiable programming for physical simulation. *arXiv preprint arXiv:1910.00935*.

Laine, S.; Hellsten, J.; Karras, T.; Seol, Y.; Lehtinen, J.; and Aila, T. 2020. Modular primitives for high-performance differentiable rendering. *ACM Transactions on Graphics (ToG)*, 39(6): 1–14.

Lavington, J. W.; Zhang, K.; Lioutas, V.; Niedoba, M.; Liu, Y.; Green, D.; Naderiparizi, S.; Liang, X.; Dabiri, S.; Ścibior, A.; et al. 2024. TorchDriveEnv: A Reinforcement Learning Benchmark for Autonomous Driving with Reactive, Realistic, and Diverse Non-Playable Characters. *arXiv preprint arXiv:2405.04491*.

Li, Q.; Peng, Z.; Feng, L.; Zhang, Q.; Xue, Z.; and Zhou, B. 2022. Metadrive: Composing diverse driving scenarios for generalizable reinforcement learning. *IEEE transactions on pattern analysis and machine intelligence*, 45(3): 3461–3475.

Li, Q.; Peng, Z. M.; Feng, L.; Liu, Z.; Duan, C.; Mo, W.; and Zhou, B. 2023a. Scenarionet: Open-source platform for large-scale traffic scenario simulation and modeling. *Advances in neural information processing systems*, 36: 3894–3920.

Li, S.; Huang, Z.; Chen, T.; Du, T.; Su, H.; Tenenbaum, J. B.; and Gan, C. 2023b. Dexdeform: Dexterous deformable object manipulation with human demonstrations and differentiable physics. *arXiv preprint arXiv:2304.03223*.

Lutter, M.; Silberbauer, J.; Watson, J.; and Peters, J. 2021. Differentiable physics models for real-world offline model-based reinforcement learning. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, 4163–4170. IEEE.

Macklin, M. 2022. Warp: A High-performance Python Framework for GPU Simulation and Graphics. https://github.com/nvidia/warp. NVIDIA GPU Technology Conference (GTC).

Martinez, M.; Sitawarin, C.; Finch, K.; Meincke, L.; Yablonski, A.; and Kornhauser, A. 2017. Beyond grand theft auto V for training, testing and enhancing deep learning in self driving cars. *arXiv preprint arXiv:1712.01397*.

Moerland, T. M.; Broekens, J.; Plaat, A.; Jonker, C. M.; et al. 2023. Model-based reinforcement learning: A survey. *Foundations and Trends® in Machine Learning*, 16(1): 1–118.

Montali, N.; Lambert, J.; Mougin, P.; Kuefler, A.; Rhinehart, N.; Li, M.; Gulino, C.; Emrich, T.; Yang, Z.; Whiteson, S.; et al. 2024. The waymo open sim agents challenge. *Advances in Neural Information Processing Systems*, 36.

Murthy, J. K.; Macklin, M.; Golemo, F.; Voleti, V.; Petrini, L.; Weiss, M.; Considine, B.; Parent-Lévesque, J.; Xie, K.; Erleben, K.; et al. 2020. gradsim: Differentiable simulation for system identification and visuomotor control. In *International conference on learning representations*.

Nachkov, A.; Paudel, D. P.; and Van Gool, L. 2025. Autonomous Vehicle Controllers From End-to-End Differentiable Simulation. In *2025 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*.

Nayakanti, N.; Al-Rfou, R.; Zhou, A.; Goel, K.; Refaat, K. S.; and Sapp, B. 2023. Wayformer: Motion forecasting via simple & efficient attention networks. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, 2980–2987. IEEE.

Newbury, R.; Collins, J.; He, K.; Pan, J.; Posner, I.; Howard, D.; and Cosgun, A. 2024. A Review of Differentiable Simulators. *IEEE Access*.

Pathak, D.; Agrawal, P.; Efros, A. A.; and Darrell, T. 2017. Curiosity-driven exploration by self-supervised prediction. In *International conference on machine learning*, 2778–2787. PMLR.

Romero, A.; Song, Y.; and Scaramuzza, D. 2024. Actor-critic model predictive control. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, 14777–14784. IEEE.

Schrittwieser, J.; Antonoglou, I.; Hubert, T.; Simonyan, K.; Sifre, L.; Schmitt, S.; Guez, A.; Lockhart, E.; Hassabis, D.; Graepel, T.; et al. 2020. Mastering atari, go, chess and shogi by planning with a learned model. *Nature*, 588(7839): 604–609.

Song, Y.; Kim, S.; and Scaramuzza, D. 2024. Learning Quadruped Locomotion Using Differentiable Simulation. *arXiv preprint arXiv:2403.14864*.

Sun, Q.; Huang, X.; Williams, B. C.; and Zhao, H. 2022. Intersim: Interactive traffic simulation via explicit relation modeling. In 2022 IEEE. In *RSJ International Conference on Intelligent Robots and Systems (IROS)*, 11416–11423.

Sutton, R. S. 1991. Dyna, an integrated architecture for learning, planning, and reacting. *ACM Sigart Bulletin*, 2(4): 160–163.

Sutton, R. S.; McAllester, D.; Singh, S.; and Mansour, Y. 1999. Policy gradient methods for reinforcement learning with function approximation. *Advances in neural information processing systems*, 12.

Vinitsky, E.; Lichtlé, N.; Yang, X.; Amos, B.; and Foerster, J. 2022. Nocturne: a scalable driving benchmark for bringing multi-agent learning one step closer to the real world. *Advances in Neural Information Processing Systems*, 35: 3962–3974.

Wang, Y.; Zhao, T.; and Yi, F. 2023. Multiverse Transformer: 1st Place Solution for Waymo Open Sim Agents Challenge 2023. *arXiv preprint arXiv:2306.11868*.

Wiedemann, N.; Wüest, V.; Loquercio, A.; Müller, M.; Floreano, D.; and Scaramuzza, D. 2023. Training efficient controllers via analytic policy gradient. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, 1349–1356. IEEE.

Xu, J.; Chen, T.; Zlokapa, L.; Foshey, M.; Matusik, W.; Sueda, S.; and Agrawal, P. 2021. An end-to-end differentiable framework for contact-aware robot design. *arXiv preprint arXiv:2107.07501*.

Xu, J.; Makoviychuk, V.; Narang, Y.; Ramos, F.; Matusik, W.; Garg, A.; and Macklin, M. 2022. Accelerated policy learning with parallel differentiable simulation. *arXiv preprint arXiv:2204.07137*.

Zhang, J.; Tai, L.; Yun, P.; Xiong, Y.; Liu, M.; Boedecker, J.; and Burgard, W. 2019. Vr-goggles for robots: Real-to-sim domain adaptation for visual control. *IEEE Robotics and Automation Letters*, 4(2): 1148–1155.

Zhang, Z.; Sakaridis, C.; and Van Gool, L. 2024. Trafficbots v1. 5: Traffic simulation via conditional vaes and transformers with relative pose encoding. *arXiv preprint arXiv:2406.10898*.

Zhou, Z.; Haibo, H.; Chen, X.; Wang, J.; Guan, N.; Wu, K.; Li, Y.-H.; Huang, Y.-K.; and Xue, C. J. 2024. Behaviorgpt: Smart agent simulation for autonomous driving with next-patch prediction. *Advances in Neural Information Processing Systems*, 37: 79597–79617.