Long-term Interactive Driving Simulation: MPC to the Rescue*

Zhengxiao Han[1,2]; Zhijie Yan[2,3]; Yang Li[2,4]; Pengfei Li[2]; Yifeng Shi[5]; Nairui Luo[5]; Xu Gao[5]; Yongliang Shi[2]; Pengfei Huang[2]; Jiangtao Gong[2]; Guyue Zhou[2]; Yilun Chen[2]; Hang Zhao[2]; Hao Zhao[2] **

[1] Beijing University of Chemical Technology
[2] Tsinghua University
[3] Beihang University
[4] University of California, San Diego
[5] Baidu, Inc.
hanzx0917@outlook.com, zhaohao@air.tsinghua.edu.cn

Abstract. Simulation now plays an important role in the development of autonomous driving algorithms as it can significantly reduce the economical cost and ethical risk of real-world testing. However, building a high-quality driving simulator is not trivial as it calls for realistic interactive behaviors of road agents. Recently, several simulators employ interactive trajectory prediction models learnt in a data-driven manner. While they are successful in generating short-term interactive scenarios, the simulator quickly breaks down when the time horizon gets longer. We identify the reason behind: existing interactive trajectory predictors suffer from the out-of-domain (OOD) problem when recursively feeding predictions as the input back to the model. To this end, we propose to introduce a tailored model predictive control (MPC) module as a rescue into the state-of-the art interactive trajectory prediction model M2I, forming a new simulator named M^{2}Sim. Notably, M^{2}Sim can effectively address the OOD problem of long-term simulation by enforcing a flexible regularization that admits the replayed data, while still enjoying the diversity of data-driven predictions. We demonstrate the superiority of M^{2}Sim using both quantitative results and visualizations and release our data, code and models: https://github.com/0nhc/m2sim.

Keywords: Autonomous Driving · Interactive Simulator · MPC.

1 Introduction

Autonomous driving [1–5] is one of the most important AI applications nowadays. Collecting data for dangerous driving scenarios is challenging, making simulation the preferred choice for algorithm development. Thus, building a high-quality driving simulator with realistic interactive behaviors is crucial.

Fig. 1-a shows a replayed data clip at an intersection, where the red line represents the future trajectory of a chosen ego car. Fig. 1-b demonstrates the

* Sponsored by Baidu Inc. through Apollo-AIR Joint Research Center.
** Corresponding Author: Hao Zhao

drawback of a non-interactive simulator. The ego car, controlled by a to-be-tested algorithm, collides with other cars following the replayed data.

To address this issue, recent works integrate interactive trajectory predictors [6–9], using neural networks to capture the diverse distribution of real-world interactive trajectories. However, they encounter the out-of-distribution (OOD) [10] problem in long-term simulation due to their recursive nature [11]. As shown in Fig. 1-c, this leads to the unrealistic long-term simulation.

To alleviate the OOD issue, we propose using model predictive control (MPC) [12–17] as a rescue mechanism (Fig. 1-d). This module introduces the replayed trajectory as a flexible regularization term. And the interactive trajectory predictor outputs serve as another fundamental regularization term, which offers diversity to the system. Other tailored add-ons like control effort and smoothness terms make the behavior of our agents more realistic.

In one word, the contribution of this study is a simulation system combining the advantages of data-driven interactive trajectory predictors and a tailored model predictive controller. To the best of our knowledge, M^{2}Sim is the first system that demonstrates realistic long-term interactive driving simulation results in the literature.

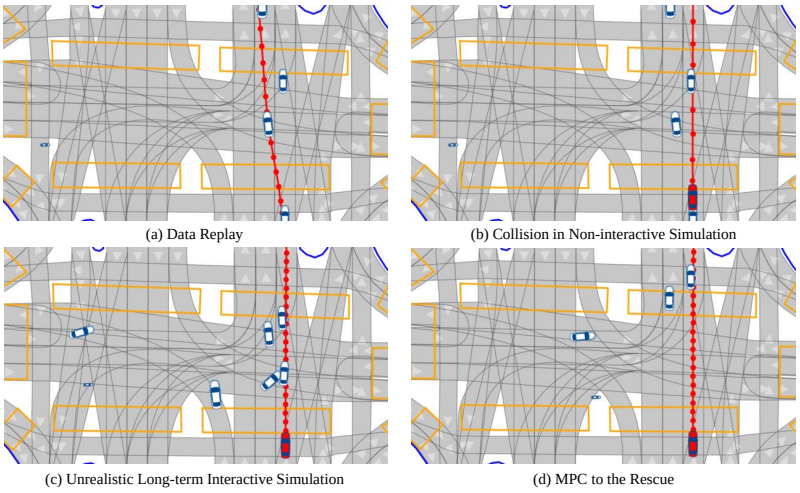

Fig. 1. (a) Data replay. (b) Collisions happen in a non-interactive simulator. (c) Other cars react according to a learned model. In the long term, their trajectories become unrealistic. (d) We leverage MPC to address the OOD issue in (c).

2 Related Work

2.1 Interactive Trajectory Prediction

For interactive trajectory prediction, some works utilize graph neural networks to learn the interaction relationship between different vehicles [18,19], and other

works use the attention in transformer to model the interaction relationship between vehicles [20, 21]. M2I [7], which is the engine of choice in our simulator for its state-of-the-art performance, uses both rasterized map representation and polyline representation, and integrates graph neural network and attention mechanism for interactive prediction. Since it utilizes future trajectories of the ego vehicle, it has good prediction accuracy and high computational efficiency. However learning-based approaches lack interpretability and may break specific physical constraints. What's worse, their behaviors become unpredictable and unreliable when the time horizon gets long.

2.2 Traffic Simulation

As for simulating interactive behaviors, autonomous driving simulators such as SUMO [22], CityFlow [23], CommonRoad [24] focus on multi-agent traffic flow simulation, but due to the lack of realistic traffic data, they cannot simulate traffic flow with interactive behaviors. Recently some works such as TrafficSim [25], SimNet [26], etc. learn from data collected in real world to model interactive multi-agent behaviors. These approaches only learn from all the agents' past behaviors at once, without considering their future interactions. InterSim [27] simplifies the prediction of future interactions into a binary classification problem, which has better interpretability on relationship prediciton and higher computational efficiency than counterparts using latent-variable [25] or cost functions [28] for to model future collisions. However when we test these learning-based approaches, we found them quickly breaking down when the time horizon of traffic scenarios gets longer, which is caused by the out-of-domain(OOD) problem [10].

3 Method

3.1 Overview

As shown in Fig. 2, our simulator M^{2}Sim addresses the OOD problem by integrating an MPC module with a data-driven interactive trajectory predictor. Here, we choose the state-of-the-art predictor M2I [7]. In the MPC formulation, we add both M2I's output trajectories and ground truth trajectories (explained later) to the optimization problem as constraints, which enforces a flexible regularization that admits the replayed ground truth while still enjoying the diversity of data-driven predictions.

3.2 M2I

As shown in Fig. 3, the trajectory predictor has three modules: relation predictor, marginal trajectory predictor, and conditional trajectory predictor. The relation predictor predicts whether each road agent will be an influencer (and yield), a reactor (and be yielded), or neither. Then the marginal trajectory predictor of it predicts the future 8 seconds of trajectories of all road agents without considering their potential interactions. Finally, with the predicted relations and marginal trajectories, its conditional trajectory predictor modifies the trajectories of reactors to account for their interactions with influencers.

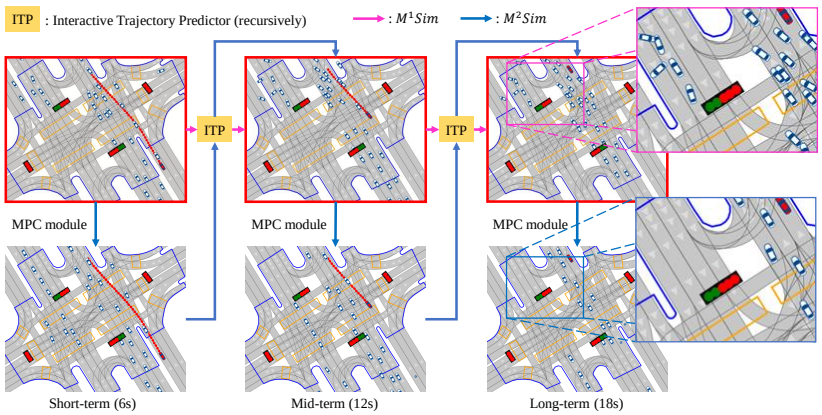

Fig. 2. The structure of M^{2}Sim. M^{1}Sim (M2I without the MPC module), denoted by the flow of pink arrows, can produce reasonable results in the short term, but its behaviors become unrealistic in the long term due to recursively feeding predictions into the model. As shown in the enlarged pink box, agents form a cluster in the long term. However M^{2}Sim (M2I with the MPC module), denoted by the flow of blue arrows, addresses the OOD problem by integrating the MPC module. Note that we conceptually use the same image to present recursively predicted results in red boxes, which are actually different for M^{1}Sim and M^{2}Sim.

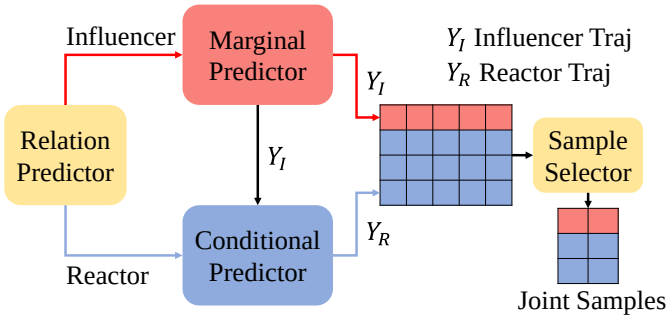

Fig. 3. A brief recap of the structure of M2I, which is our data-driven interactive trajectory prediction engine. It ourputs several trajectories with confidence values, so that we can randomly sample from them to generate diverse trajectories.

In addition to the three modules for predicting relations and trajectories, it laso has a module for selecting trajectories. Specifically, for N single vehicle trajectories, we have N^2 pairs of interactions. The sample selector outputs the top K out of the N^2 pairs based on the ranking of confidence scores. We can generate diverse trajectories with interactive trajectory predictor by selecting trajectories randomly according to the confidence values.

The interactive trajectory predictor takes 1.1 seconds of past trajectories and predicts 8 seconds of future trajectories (represented as a set of coordinates x, y). By differential operation, we can get yaw and v from discrete coordinates x, y of future trajectories. The final output trajectory can be described as $z_{ref} = [x, y, yaw, v]$ denoting [coordinate x, coordinate y, heading angle, speed].

3.3 MPC

As shown in Fig. 4, we design an MPC controller considering both M2I's output trajectories and ground truth trajectories as constraints for the optimization problem. In addition, MPC comes with a kinematic model, which also functions as a constraint and ensures the physical plausibility of its output trajectories.

We denote an autonomous vehicle(AV)'s state of [coordinate x, coordinate y, heading angle, speed] by $z(\tau) = [x(\tau), y(\tau), yaw(\tau), v(\tau)]$, and its control input of [acceleration, steering angle] by $u(\tau) = [a(\tau), \delta(\tau)]$. Then we denote the prediction horizon by $T_p \in \mathbb{N}$. Thus, the problem can be defined as solving the optimal control input $u(\tau) = [a(\tau), \delta(\tau)]$ at specific time τ. Our MPC formulations are as follows:

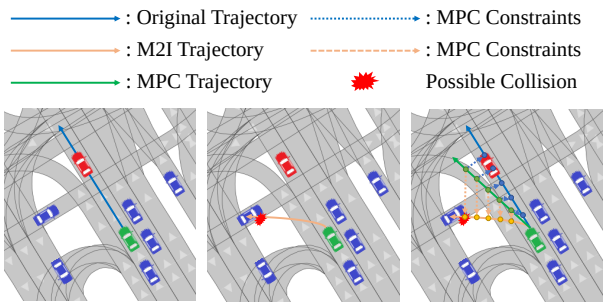

Fig. 4. The MPC module considers both the original replay trajectory and the predicted interactive trajectory as constraints to the optimization problem.

Optimization Problem Setup At time τ, we denote control inputs of $U(\tau) \in U_b \subset \mathbb{R}^{2T_P}$ and corresponding states of $Z(\tau) \in Z_b \subset \mathbb{R}^{4T_p}$:

$$U(\tau) = [u(\tau), ..., u(\tau + T_p - 1)]^T, \quad U_b = [u_{min},\ u_{max}]$$
$$Z(\tau) = [z(\tau + 1), ..., z(\tau + T_p)]^T, \quad Z_b = [z_{min},\ z_{max}]$$

Our MPC module considers both M2I's output trajectories and ground truth trajectories as inputs, while taking smoothness and control efforts into account. Let $\|\cdot\|$denotes Euclidean Norm, we design the following cost function $J(Z(\tau),\, U(\tau))$:

$$J = \sum_{t=\tau+1}^{\tau+T_p} \lambda_{z1}\|z(t) - z_{ref1}(t)\|^2 + \sum_{t=\tau+1}^{\tau+T_p} \lambda_{z2}\|z(t) - z_{ref2}(t)\|^2 \quad \text{(State Error)}$$
$$+ \sum_{t=\tau}^{\tau+T_p-1} \lambda_u\|u(t)\|^2 \quad \text{(Control Effort)}$$
$$+ \sum_{t=\tau}^{\tau+T_p-1} \lambda_{\Delta u}\|u(t) - u(t-1)\|^2, \quad \text{(Smoothness)}$$

where z_{ref1} and z_{ref2} denotes trajectories provided by M2I and ground truth trajectories respectively, and all λ denotes constant values corresponding to their weights in the cost function.

Thus, the optimization problem can be defined as solving:

$$U^* \triangleq \underset{U}{arg\,min}\ J(Z(\tau), U(\tau))$$

System Modeling Motivated by [29], the car can be defined as a bicycle model with state $z(\tau)$ and control input $u(\tau)$ at specific time τ. Based on the bicycle model's kinematics, we can get the differential equation of z and u:

$$\dot{z} = \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{yaw} \\ \dot{v} \end{bmatrix} = \begin{bmatrix} v \times cos(yaw) \\ v \times sin(yaw) \\ \frac{v \times tan(\delta)}{l} \\ a \end{bmatrix},$$

where l is the distance between the front and rear wheels of the vehicle. Then we linearize and discretize the kinematic model to obtain the equality constraint for the optimization problem. Suppose at time τ, we observed states $\hat{z}(\tau)$ and control inputs $\hat{u}(\tau)$ of the ego vehicle, then we can infer its future state:

$$\hat{z}(\tau+1) = A \times \hat{z}(\tau) + B \times \hat{u}(\tau) + C,$$

where A, B and C are:

$$A = \begin{bmatrix} 1 & 0 & -v \times sin(yaw) \times dt & cos(yaw) \times dt \\ 0 & 1 & v \times cos(yaw) \times dt & sin(yaw) \times dt \\ 0 & 0 & 1 & \frac{tan(\delta) \times dt}{l} \\ 0 & 0 & 0 & 1 \end{bmatrix},$$

$$B = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & \frac{v}{l \times cos^2(\delta)} \\ 1 & 0 \end{bmatrix}, C = \begin{bmatrix} v \times sin(yaw) \times yaw \times dt \\ -v \times cos(yaw) \times yaw \times dt \\ -\frac{v \times \delta}{l \times cos^2(\delta)} \times dt \\ 0 \end{bmatrix}$$

At last the optimization problem can be solved with all these constraints. We use a off-the-shelf toolbox cvxpy to get optimal control inputs.

3.4 Collision Avoidance

To reduce collisions, integrating inequality constraints into the optimization problem is a natural choice. Inequality constraints such as the Euclidean distance between the ego vehicle and its surrounding vehicles may be useful for collision avoidance [30]. However, too many inequality constraints can cause high computational cost in optimization.

Thus, we designed a collision avoidance mechanism outside the optimization process by only changing the acceleration of control inputs. Suppose we have the ego vehicle of state $z(\tau) = [x(\tau), y(\tau), yaw(\tau), v(\tau)]$ and surrounding vehicles of states $z_i(\tau) = [x_i(\tau), y_i(\tau), yaw_i(\tau), v_i(\tau)], i \in N$, corresponding to [coordinate xy, heading angle, speed] at time τ, then we can get the final control inputs $u(\tau) = [a(\tau), \delta(\tau)]$ corresponding to [acceleration, steering angle] through Algorithm 1.

Algorithm 1: Framework of Collision Avoidance.

Input: Ego Vehicle of State: $z(\tau)$; Number of Surrounding Vehicles N;
Surrounding Vehicles of States: $z_i(\tau)$ where $i \in N$; Safety Distance D;
Coordinate Vector $\vec{s}$; Constant Value λ_f

Output: Control Inputs $u(\tau)$ of the Ego Vehicle

Get $z(\tau)$, $z_i(\tau), i \in N$ at time τ; $i = 0$; $\vec{s} = [0, 0]$; **foreach** i *in* N **do**
  Calculate the Euclidean distance d_i between the
  ego vehicle and the surrounding vehicle:
  $d_i = \sqrt{[x(\tau) - x_i(\tau)]^2 + [y(\tau) - y_i(\tau)]^2}$
  if $d_i < D$ **then**
    Sum up force vector $\vec{s}$.
    $\vec{s_i} = [x_i(\tau), \vec{y_i(\tau)}] - [x(\tau), \vec{y(\tau)}]$
    $\vec{s} = \vec{s} + \vec{s_i}$

The unit vector $\vec{e}$ of the ego vehicle:
$\vec{e} = v(\tau) \cdot [cos(yaw(\tau)), sin(yaw(\tau))]$
The projection p of $\vec{s}$ in the direction of $\vec{e}$: $p = \frac{\lambda_f}{\vec{s} \cdot \vec{e}}$
if $p < 0$ **then**
  Only consider the case of deceleration.
  $a(\tau) = a(\tau) + p$
  Limit $a(\tau)$ within its boundary.
  $a(\tau) = max(a(\tau), a_{min})$
Return $u(\tau) = [a(\tau), \delta(\tau)]$

4 Experiment

In this section, we introduce model details, and provide quantitative results and qualitative examples of different simulators to demonstrate the performance of M^{2}Sim.

4.1 Model Details

We used the original M2I model trained on Waymo Open Motion Dataset (WOMD) from our baseline InterSim for evaluation.

As for the MPC and collision avoidance module, we initialize default constant values of the optimization problem $U^* \triangleq arg\ min_U\ J(Z(\tau), U(\tau))$ where $\lambda_{z1} = \lambda_{z2} = 0.5$, $\lambda_u = \lambda_{\Delta u} = 1.0$. And in the collision avoidance algorithm, we initialize safety distance with $D = 8$ and $\lambda_f = 1.0$. We tuned these hyperparameters on a small validation set.

4.2 Simulation Task

The simulation task is to test the performance of agents in M^{2}Sim and our baseline InterSim [27] on driving scenarios generated by editing the data replay clips to make collisions happen (as ground truth trajectories). The editing process involves selecting a random car as the ego vehicle, generate a random trajectory for it using the marginal predictor in Fig. 3.

The reason why we choose InterSim is, InterSim uses the state-of-the-art interactive prediction model M2I, and the difference between InterSim and M^{2}Sim is, M^{2}Sim has an MPC and collision avoidance module in addition to the M2I model. Because InterSim only uses an M2I model for predicting trajectories, while M^{2}Sim leverages both an M2I model and an MPC module, we call our re-implemented InterSim as M^{1}Sim.

4.3 Quantitative Results

Method	minADE ↓	minFDE ↓	missRate ↓	mAP ↑
M^{1}Sim (re-implemented InterSim)	9.349	25.917	0.906	0.004
M^{2}Sim	**1.089**	**2.161**	**0.154**	**0.175**

Table 1. Performance of interactive prediction.

Motivated by [7, 31], we use these metrics to evaluate the performance of simulators, including:

- **minADE** (Minimum Average Displacement Error) The minADE metric computes the mean of the L2 norm between the ground truth future trajectory and the closest predicted output trajectory from M2I out of $K = 6$ (number of M2I's outputs) samples.

- **minFDE** (Minimum Final Displacement Error) The minFDE is the same as minADE to compute the displacement error, but the minFDE only computes the displacement of the final positions of ground truth trajectory and M2I's predicted trajectory.
- **missRate** (Miss Rate) A **miss** is defined as the state when none of the individual K predictions for an object are within a given lateral and longitudinal threshold of the ground truth trajectory. The missRate is calculated as the total number of misses divided by $K = 6$ (number of M2I's outputs) for M2I.
- **mAP** (Mean Average Precision) The mAP computes the area under the precision-recall curve of the prediction samples by applying confidence score thresholds.

Method	Agent-agent ↓	Agent-environment ↓
M^1Sim (re-implemented InterSim)	0.343	0.447
M^2Sim	**0.075**	**0.326**

Table 2. Different types of collision rate for M^1Sim and M^2Sim.

The results are summarized in TABLE I representing how the simulators' behaviors adhere to the ground truth data replay. We observed that M^2Sim achieves the lowest errors, the lowest miss rate and the highest precision in predicting trajectories. InterSim, also utilizing an M2I model, suffers from the out-of-domain (OOD) problem in our long-term experiments. As the simulation time gets longer, environmental vehicles in InterSim will begin to deviate from ground truth trajectories, collide with each other, and even drive out of the lane. But we add an MPC controller along with a collision avoidance module to process M2I's output trajectories, so that vehicles' trajectories are constrained to be as close as possible to ground truth trajectories.

4.4 Qualitative Examples

In Fig. 5, we present two representative scenarios to show our method's performance of long-term simulation. We present M^2Sim without MPC module as M^1Sim. In both scenarios, M^1Sim suffers from the out-of-domain (OOD) problem when recursively feeding predictions as the input back to the model, failing to keep environmental vehicles from tracking ground truth trajectories and avoiding collisions over time. But M^2Sim achieves to keep track of environmental vehicles' original trajectories and avoid collisions by adding an MPC and collision avoidance module with original trajectories as optimization constraints.

4.5 Ablation Study

We present ablation study on the MPC controller along with the collision avoidance module by comparing the collision rate of M^2Sim and M^1Sim for the same simulation task. The results are summarized in TABLE II. The agent-agent collision rate computes the number of colliding agent pairs divided by the number

of simulated agents. Similarly, the agent-environment collision rate computes the number of agents colliding with lanes or pedestrians divided by the number of simulated agents. It is not surprising to see M^{2}Sim achieves better performance due to the constraints of following ground truth trajectories and collision avoidance mechanism.

5 Conclusion

In conclusion, we present an interactive traffic simulator with an advanced interactive trajectory predictor (M2I) and address the OOD problem in long-term simulation. M^{2}Sim improved its performance by adding an MPC controller along with a collision avoidance module. In the experiments, we test M^{2}Sim and InterSim on the same simulation task to demonstrate the superiority of our M^{2}Sim. In the ablation study, we show the effectiveness of our proposed MPC module. **Limitations.** However, to avoid too much computational consumption, our proposed MPC module does not consider obstacle avoidance in the cost function. Further research could go deeper in optimization.

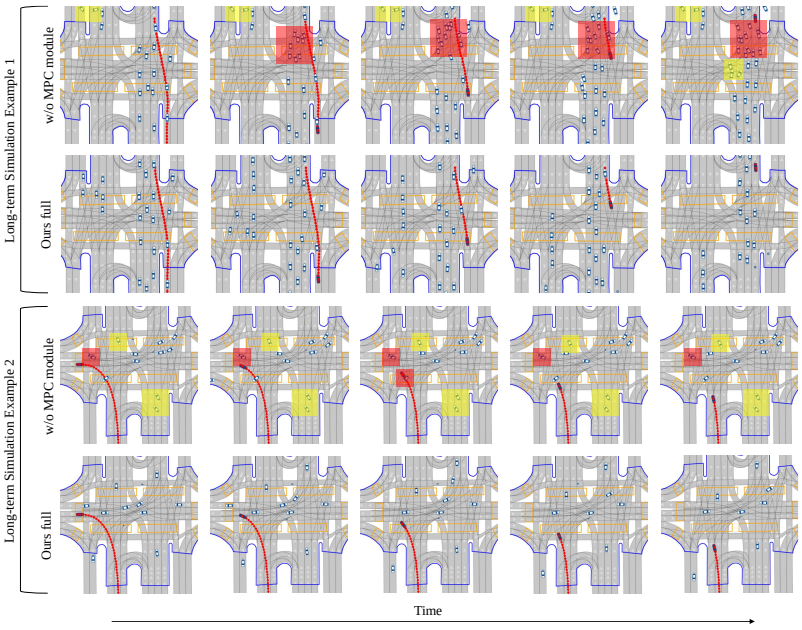

Fig. 5. Examples of M^{2}Sim successfully keeping environmental vehicles from tracking their original trajectories and avoiding collisions. Failures of keeping original trajectories with weird heading angles (the OOD problem) are highlighted in yellow boxes. Collisions are highlighted in red boxes.

References

1. B. Tian, M. Liu, H.-a. Gao, P. Li, H. Zhao, and G. Zhou, "Unsupervised road anomaly detection with language anchors," in *2023 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2023, pp. 7778–7785.
2. P. Li, R. Zhao, Y. Shi, H. Zhao, J. Yuan, G. Zhou, and Y.-Q. Zhang, "Lode: Locally conditioned eikonal implicit scene completion from sparse lidar," *arXiv preprint arXiv:2302.14052*, 2023.
3. B. Jin, X. Liu, Y. Zheng, P. Li, H. Zhao, T. Zhang, Y. Zheng, G. Zhou, and J. Liu, "Adapt: Action-aware driving caption transformer," *arXiv preprint arXiv:2302.00673*, 2023.
4. Y. Zheng, C. Zhong, P. Li, H.-a. Gao, Y. Zheng, B. Jin, L. Wang, H. Zhao, G. Zhou, Q. Zhang *et al.*, "Steps: Joint self-supervised nighttime image enhancement and depth estimation," *arXiv preprint arXiv:2302.01334*, 2023.
5. Y. Hu, J. Yang, L. Chen, K. Li, C. Sima, X. Zhu, S. Chai, S. Du, T. Lin, W. Wang *et al.*, "Planning-oriented autonomous driving," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023, pp. 17 853–17 862.
6. J. Ngiam, V. Vasudevan, B. Caine, Z. Zhang, H.-T. L. Chiang, J. Ling, R. Roelofs, A. Bewley, C. Liu, A. Venugopal *et al.*, "Scene transformer: A unified architecture for predicting future trajectories of multiple agents," in *International Conference on Learning Representations*, 2022.
7. Q. Sun, X. Huang, J. Gu, B. C. Williams, and H. Zhao, "M2i: From factored marginal trajectory prediction to interactive prediction," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 6543–6552.
8. X. Liu, Y. Wang, K. Jiang, Z. Zhou, K. Nam, and C. Yin, "Interactive trajectory prediction using a driving risk map-integrated deep learning method for surrounding vehicles on highways," *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 10, pp. 19 076–19 087, 2022.
9. K. Zhang, L. Zhao, C. Dong, L. Wu, and L. Zheng, "Ai-tp: Attention-based interaction-aware trajectory prediction for autonomous driving," *IEEE Transactions on Intelligent Vehicles*, 2022.
10. Z. Shen, J. Liu, Y. He, X. Zhang, R. Xu, H. Yu, and P. Cui, "Towards out-of-distribution generalization: A survey," *arXiv preprint arXiv:2108.13624*, 2021.
11. A. Filos, P. Tigkas, R. McAllister, N. Rhinehart, S. Levine, and Y. Gal, "Can autonomous vehicles identify, recover from, and adapt to distribution shifts?" in *International Conference on Machine Learning*. PMLR, 2020, pp. 3145–3153.
12. E. C. Kerrigan, "Predictive control for linear and hybrid systems [bookshelf]," *IEEE Control Systems Magazine*, vol. 38, no. 2, pp. 94–96, 2018.
13. P. Falcone *et al.*, "Nonlinear model predictive control for autonomous vehicles," 2007.
14. A. Carvalho, Y. Gao, A. Gray, H. E. Tseng, and F. Borrelli, "Predictive control of an autonomous ground vehicle using an iterative linearization approach," in *16th International IEEE conference on intelligent transportation systems (ITSC 2013)*. IEEE, 2013, pp. 2335–2340.
15. Y. Gao, T. Lin, F. Borrelli, E. Tseng, and D. Hrovat, "Predictive control of autonomous ground vehicles with obstacle avoidance on slippery roads," in *Dynamic systems and control conference*, vol. 44175, 2010, pp. 265–272.
16. C. E. Beal and J. C. Gerdes, "Model predictive control for vehicle stabilization at the limits of handling," *IEEE Transactions on Control Systems Technology*, vol. 21, no. 4, pp. 1258–1269, 2012.

17. J. Levinson, J. Askeland, J. Becker, J. Dolson, D. Held, S. Kammel, J. Z. Kolter, D. Langer, O. Pink, V. Pratt *et al.*, "Towards fully autonomous driving: Systems and algorithms," in *2011 IEEE intelligent vehicles symposium (IV)*. IEEE, 2011, pp. 163–168.
18. A. Mohamed, K. Qian, M. Elhoseiny, and C. Claudel, "Social-stgcnn: A social spatio-temporal graph convolutional neural network for human trajectory prediction," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2020, pp. 14 424–14 432.
19. S. Casas, C. Gulino, S. Suo, K. Luo, R. Liao, and R. Urtasun, "Implicit latent variable model for scene-consistent motion forecasting," in *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XXIII 16*. Springer, 2020, pp. 624–641.
20. N. Kamra, H. Zhu, D. K. Trivedi, M. Zhang, and Y. Liu, "Multi-agent trajectory prediction with fuzzy query attention," *Advances in Neural Information Processing Systems*, vol. 33, pp. 22 530–22 541, 2020.
21. N. Nayakanti, R. Al-Rfou, A. Zhou, K. Goel, K. S. Refaat, and B. Sapp, "Wayformer: Motion forecasting via simple & efficient attention networks," *arXiv preprint arXiv:2207.05844*, 2022.
22. P. A. Lopez, M. Behrisch, L. Bieker-Walz, J. Erdmann, Y.-P. Flötteröd, R. Hilbrich, L. Lücken, J. Rummel, P. Wagner, and E. Wießner, "Microscopic traffic simulation using sumo," in *2018 21st international conference on intelligent transportation systems (ITSC)*. IEEE, 2018, pp. 2575–2582.
23. H. Zhang, S. Feng, C. Liu, Y. Ding, Y. Zhu, Z. Zhou, W. Zhang, Y. Yu, H. Jin, and Z. Li, "Cityflow: A multi-agent reinforcement learning environment for large scale city traffic scenario," in *The world wide web conference*, 2019, pp. 3620–3624.
24. M. Althoff, M. Koschi, and S. Manzinger, "Commonroad: Composable benchmarks for motion planning on roads," in *2017 IEEE Intelligent Vehicles Symposium (IV)*. IEEE, 2017, pp. 719–726.
25. S. Suo, S. Regalado, S. Casas, and R. Urtasun, "Trafficsim: Learning to simulate realistic multi-agent behaviors," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 10 400–10 409.
26. L. Bergamini, Y. Ye, O. Scheel, L. Chen, C. Hu, L. Del Pero, B. Osiński, H. Grimmett, and P. Ondruska, "Simnet: Learning reactive self-driving simulations from real-world observations," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2021, pp. 5119–5125.
27. Q. Sun, X. Huang, B. C. Williams, and H. Zhao, "Intersim: Interactive traffic simulation via explicit relation modeling," in *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2022, pp. 11 416–11 423.
28. J. Zhou, R. Wang, X. Liu, Y. Jiang, S. Jiang, J. Tao, J. Miao, and S. Song, "Exploring imitation learning for autonomous driving with feedback synthesizer and differentiable rasterization," in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2021, pp. 1450–1457.
29. J. Kong, M. Pfeiffer, G. Schildbach, and F. Borrelli, "Kinematic and dynamic vehicle models for autonomous driving control design," in *2015 IEEE intelligent vehicles symposium (IV)*. IEEE, 2015, pp. 1094–1099.
30. M. Cho, Y. Lee, and K.-S. Kim, "Model predictive control of autonomous vehicles with integrated barriers using occupancy grid maps," *IEEE Robotics and Automation Letters*, 2023.
31. T. Kühner and J. Kümmerle, "Large-scale volumetric scene reconstruction using lidar," in *2020 IEEE international conference on robotics and automation (ICRA)*. IEEE, 2020, pp. 6261–6267.