A SIMPLE YET EFFECTIVE METHOD FOR SIMULATING REALISTIC MULTI-AGENT BEHAVIORS

Cheng Qian*
College of Information Science and Engineering
Northeastern University
Shenyang, China
qiancheng625@gmail.com

Di Xiu*
School of Electronic, Electrical and Communication Engineering
University of Chinese Academy of Sciences
Beijing, China
suoliweng98@gmail.com

Minghao Tian
DiDi Autonomous Driving
Beijing, China
tianminghao@didiglobal.com

ABSTRACT

In this technical report, we present the 2nd place solution of 2023 Waymo Open Sim Agents Challenge (WOSAC) [4]. We propose a simple yet effective autoregressive method for simulating multi-agent behaviors, which is built upon a well-known multimodal motion forecasting framework called Motion Transformer (MTR) [5] with proper post-processing module applied. Our submission named *MTR+++* achieves 0.4697 on the Realism Meta metric on the public WOSAC leaderboard.

1 Introduction

How to validate the safety of an autonomous driving systems (ADS) is an important topic in both research community and L2+ autonomous driving industry. In UNECE document 'Proposal for the Future Certification of Automated/Autonomous Driving Systems' (ECE/TRANS/WP.29/GRVA/2019/13), the authors proposed the so-called three pillars of safety validation: simulation, closed-road scenario test and real-world road test. Simulation is undoubtedly the safest and the most cost-efficient 'pillar'. However, simulating traffic participants in a realistic way is very challenging [2].

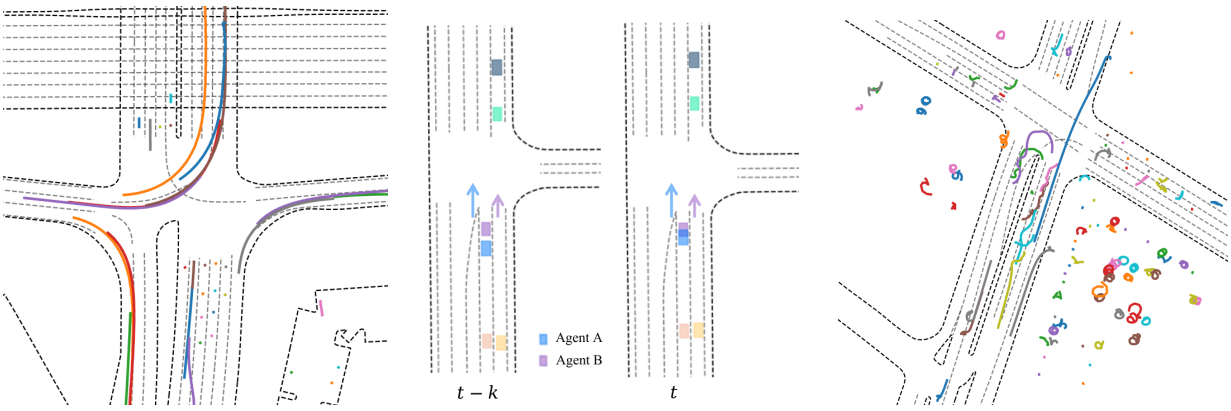
WOSAC [4] is a new challenge focusing on realistically simulating traffic agents in an interactive way. Contestants are asked to generate the future 8-second behaviors of the agents in an autoregressive pattern, given the last 1-second historical motion data together with the map features of the scenario. The challenge is actually very similar to motion forecasting in literature [1, 3, 6]. We adapt one of the methods called Motion Transformer (MTR) [5] to help us generate the possible futures.

Motion Transformer (MTR) framework. Motion Transformer (MTR) framework is a transformer encoder-decoder model which achieved the SOTA performance on both the marginal and joint motion prediction benchmarks in 2022 Waymo Open Dataset Challenges. It uses vectorized representation to handle both input trajectories and road map as poly-lines, and adopts the agent-centric strategy that organizes all inputs to the local coordinate system centered at that agent. During inference, the input of the MTR model is the historical motion data $\{[cx, cy, cz, dx, dy, dz, \theta, vel_x, vel_y, valid]_i\}$ of all the traffic participants (the local map features are included by collecting the map poly-lines nearby for each agent), where $[cx, cy, cz]$ represent the location; $[dx, dy, dz, \theta]$ are the length, width, height and heading, respectively; the $[vel_x, vel_y]$ are the x, y-direction velocity and the $valid$ is a dataset-related feature which represents whether the groundtruth provided is valid. The output of the model are the predicted trajectories in form of $\{[cx, cy, vel_x, vel_y]_i\}$. We run the same inference procedure for each agent in one inference round.

2 Method

We propose a simple method to simulate the behaviors of the agents given the past historical motion data. MTR framework is used as the motion forecasting block to produce a hybrid of open-loop and closed-loop motion data by executing model inference autoregressively at 0.5Hz (more

*This work was done during internship at DiDi Autonomous Driving



(a) (b) (c)

Figure 1: (a) Trajectories directly output by MTR model; (b) Two vehicles collide after adapting the trajectories with the highest probabilities generated by MTR model; (c) Scenario generated at 10Hz;

discussion on the update rate can be found in Section 3.2). One inference round includes 6 possible future trajectories generated for each agent (with sum-to-one probabilities assigned). At any simulation step, agents utilize MTR-generated trajectories and conduct the collision-mitigation policy (introduced in Section 2.2) to pick 1 out of 6. To be more specific, MTR model together with the collision-mitigation policy infers a 2-second $[0s, 2s]$ future trajectory for each agent , based on the past 1-second historical motion data alongside the map features. After applying heading calculation (introduced in Section 2.1) to the trajectories, the $[1s, 2s]$ halves of the trajectories will be used as the input of the MTR model at the next simulation step. During the entire simulation, we keep the altitude, the length, the width and the height of each agent as constants (read from historical motion data). After running several rounds, a simulated scenario will be generated. To generate 32 results as required by the challenge, we simply select 32 variants of the fine-tuned MTR models and each generates one. More details can be found in Section 3.1. This method achieves 0.4697 on the Realism Meta metric, which is the 2rd place of 2023 WOSAC.

2.1 Heading calculation

Since MTR generates high-quality trajectories which is showed empirically in previous work (Figure 1(a)), we simply use the (x, y) coordinates of the generated trajectories to compute the corresponding headings of the vehicles and cyclists (implemented using *np.arctan2* to calculate the heading). For those agents predicted to hold still (which can be induced from their trajectories by a threshold[2]), we keep their heading still. For those agents whose predicted headings distort much, we stabilize the headings by checking the consecutive ones. If the difference between two consecutive headings $[t, t + 0.1s]$ is larger than 0.3rad, we overwrite the heading at $t + 0.1s$ and force it to be equal to the heading at t. A further normalization step is needed here to make sure $h_i \in [-\pi, \pi)$.

2.2 Collision-mitigation policy

Suppose there are N agents (including the ADV agent) to be simulated in a given scene. With 6 trajectories generated for each agent by MTR, there could be 6^N possible results at each update step. We observed that simply picking the trajectories with the largest probability could lead to scenarios with unrealistic collisions (Figure 1(b)). To mitigate the presence of collisions while avoiding brute-force searching, we propose a greedy strategy to try to select one combination with reasonable small collision counts. The idea is to first build the $6N$ by $6N$ distance matrix D where entry $D_{6m+i-1,6n+j-1}$ indicates the minimum L^2 distance between the ith-highest trajectories of agent m and the jth one of agent n, where $0 \leq m, n \leq N-1$ and $1 \leq i, j \leq 6$. If $D_{6m+i-1,6n+j-1} \leq (width_m + width_n)/2$ where $width_k$ is the width of agent k, then adapting these two trajectories will definitely lead to a collision. Thus, we can build a 0-1 matrix C where $C_{6m+i-1,6n+j-1} = 0$ if a collision happens between the ith trajectories of agent m and the jth of agent n otherwise it will be set to 1. Since trajectories generated for the same agent are all originated from the same starting position, these trajectories collide and thus we have N 6×6 zero-blocks lying on the diagonal of C. The matrix C is symmetric and thus can be viewed as an adjacent matrix of an undirected graph with $6N$ nodes (keep the same indexing of the matrix). We then run Algorithm 1 to find a dense subgraph of size N[3]. The

[2]Any agent with moving distance less than 0.3m during 2s according to its predicted trajectory is treated as a stopped agent

[3]Under the adjacency matrix setting, the density of a subgraph of size l can be easily calculated by the sum of all the entries of the corresponding submatrix divided by $l \times (l-1)$. A clique is a subgraph with density equal to 1. Here, we say a subgraph is dense if its density is ≥ 0.95.

denser the subgraph is, the less the collision count will be. Finally, we select the corresponding trajectories according to the dense subgraph[4].

Algorithm 1: Clique/dense subgraph-finding heuristic

Input: C — the $(6N \times 6N)$-adjacency matrix described in Section 2.2;
Output: $\boldsymbol{c}$ — an array of indices of size N indicating the vertices of the derived dense subgraph;
1. Initialize $\boldsymbol{c} := [0, 6, \cdots, 6(N-1)]$;
2. Check whether $\{\boldsymbol{c}[0], \cdots, \boldsymbol{c}[N-1]\}$ forms a clique; if so, stop and OUTPUT $\boldsymbol{c}$;
3. Compute deg the degree array of vertices;
4. Define two recursive functions $FindDSG(i, l, s)$ and $FindClique(i, l, s)$ which gradually expands the subgraph. Here, i indicates the starting index of the search for the lth joiner vertex, expecting to find one with degree at least $(s-1)$:

DEF $FindDSG(i, l, s)$:
  FOR $j \in [i, i+5]$:
    IF $deg[j] \geq s-1$:
      SET $\boldsymbol{c}[l] = j$;
      IF $\{\boldsymbol{c}[0], \cdots, \boldsymbol{c}[l]\}$ forms a dense subgraph:
        IF $l < s-1$:
          RUN $FindDSG\left(6 \times (l+1), l+1, s\right)$;
        ELSE stop and OUTPUT $\boldsymbol{c}$;

DEF $FindClique(i, l, s)$:
  LET $j_{tmp} := -1$;
  FOR $j \in [i, i+5]$:
    SET $\boldsymbol{c}[l] := j$;
    IF $\{\boldsymbol{c}[0], \cdots, \boldsymbol{c}[l]\}$ forms a clique:
      SET $j_{tmp} := j$ and BREAK;
  IF $j_{tmp} \neq -1$:
    SET $\boldsymbol{c}[l] := j_{tmp}$;
    IF $l < s-1$:
      RUN $FindClique\left(6 \times (l+1), l+1, s\right)$;
    ELSE stop and OUTPUT $\boldsymbol{c}$;
  ELSE run $FindDSG(i, l, s)$;

5. RUN $FindClique(0, 0, N)$

3 Experiments

Due to time constraints, all the evaluations in this section were conducted only on the first TFRecord file of WOMD v1.2, which consists of 287 validation scenarios.

3.1 Implementation Details

We trained the MTR model with the same hyperparameter settings as in [5] on both WOMD v1.1 and v1.2 except for the learning rate. Two rounds of pre-training were conducted based on WOMD v1.1 to get two baseline models (each ran 30 epochs with 8 GPUs (NVIDIA RTX A6000) and batch size of 128 scenes): the learning rate is decayed by a factor of 0.5 every 2 epochs from epoch 20 and epoch 22, respectively. These two pretrained models were further fine-tuned on WOMD v1.2 for 20 epochs. Therefore, we obtained 40 separate models, one for each of 40 epochs as candidates. Finally, we picked 32 models out of 40 with the highest mAP score, and applied them to generate scenarios in the autoregressive fashion introduced in Section 2. Using models from the results of different epochs can not only be viewed as an ensemble method, but also an implicit way of adding noise which can generate more diverse multi-agent behaviors. Experiments also showed that the more models we included in the ensemble, the higher the overall composite metric would be (see Section 3.4 below).

3.2 Update rate

We attempted to use a frequency of 10Hz for autoregressive simulation, which resulted in poor performance (Table 1). After checking the visualization (Figure 1(c)), we observed that several generated trajectories were unrealistic like wandering on the map without specific purpose. This may be because each simulation step would bring a certain disturbance error to the predicted trend of the previous simulation step. The more simulation steps we conduct, the greater the accumulated error will be.

3.3 Effects of collision-mitigation policy

We compare our collision-mitigation update policy with the update policy which always selects the trajectories with the highest probability scores. Table 2 shows that our proposed collision-mitigation policy slightly improves the quality of the simulated scenarios. Interestingly, it improves all the metrics listed a little bit.

3.4 Number of MTR variants

Since our model runs at 0.5Hz, we have 4 simulation steps in total in 8s simulation duration. We use notation $[m_1, m_2, m_3, m_4]$ to denote the number of different scenarios generated by our model at each step. For example, $[1, 2, 4, 4]$ means that we have generated 2 different scenarios at the second simulation step by picking the top-2 combinations of trajectories in the sense of the joint probability. We tested on $1, 2, 4, 8, 16, 32$ variants respectively with heading calculation module and collision-mitigation policy enabled. Empirical results (Table 3) showed that 32 variants are indeed needed and it significantly improves the Interactive metrics.

3.5 An end-to-end model MTR_E.

We also propose an end-to-end variant of MTR, called MTR_E, where can directly predict the $[cx, cy, cz, \theta, vel_x, vel_y]$ of the agents. Specifically, we have simply modified the decoder section of MTR. We

[4] If the vertex $(6m + i - 1)$ is included in the subgraph, then we pick the ith-highest trajectory of agent m

Table 1: Different update rate.

Update rate (repeated 32 times)	**Realism Meta metric** ↑	Kinematic metrics ↑	Interactive metrics ↑	Map-based metrics ↑	minADE ↓
10Hz	0.2836	0.0934	0.1243	0.0659	8.9345
0.5Hz	**0.4311**	**0.1232**	**0.1554**	**0.1525**	**2.2093**

Table 2: Effects of collision-mitigation policy.

Collision-mitigation policy	**Realism Meta metric** ↑	Kinematic metrics ↑	Interactive metrics ↑	Map-based metrics ↑	minADE ↓
×	0.4743	0.1298	0.1747	0.1698	1.7560
✓	**0.4753**	**0.1299**	**0.1754**	**0.1700**	**1.7554**

Table 3: Effects of number of variants of MTR model selected.

number of variants	**Realism Meta metric** ↑	Kinematic metrics ↑	Interactive metrics ↑	Map-based metrics ↑	minADE ↓
1 (repeat 32 times)	0.4336	0.1228	0.1580	0.1528	2.1848
1 [1,2,4,4]	0.4398	0.1194	0.1593	0.1611	2.1674
2 [1,1,4,4]	0.4487	0.1209	0.1646	0.1632	2.1373
4 [1,1,2,4]	0.4519	0.1221	0.1659	0.1639	2.0618
8 [1,1,1,4]	0.4652	0.1286	0.1690	0.1675	1.9015
16 [1,1,1,2]	0.4712	0.1288	0.1732	0.1692	1.8096
32 [1,1,1,1] (**Submitted**)	0.4753	0.1299	0.1754	0.1700	1.7554
32 [1,1,1,1] (MTR_E)	**0.4958**	**0.1484**	**0.1738**	**0.1735**	**1.6642**

changed the output dimension of MLP in dense future prediction module and motion prediction module while adding extra $L1$ losses for cz and θ to the total loss. Table 3 also shows the MTR_E improves all the metrics listed, which leads to a 4.3% increase on Realism Meta metric than our submission MTR+++.

References

[1] Y. Chai, B. Sapp, M. Bansal, and D. Anguelov. Multipath: Multiple probabilistic anchor trajectory hypotheses for behavior prediction. *arXiv preprint arXiv:1910.05449*, 2019.

[2] Q. Chao, H. Bi, W. Li, T. Mao, Z. Wang, M. C. Lin, and Z. Deng. A survey on visual traffic simulation: Models, evaluations, and applications in autonomous driving. *Computer Graphics Forum*, 39(1):287–308, 2020.

[3] W. Luo, C. Park, A. Cornman, B. Sapp, and D. Anguelov. Jfp: Joint future prediction with interactive multi-agent modeling for autonomous driving. In K. Liu, D. Kulic, and J. Ichnowski, editors, *Proceedings of The 6th Conference on Robot Learning*, volume 205 of *Proceedings of Machine Learning Research*, pages 1457–1467. PMLR, 14–18 Dec 2023.

[4] N. Montali, J. Lambert, P. Mougin, A. Kuefler, N. Rhinehart, M. Li, C. Gulino, T. Emrich, Z. Yang, S. Whiteson, B. White, and D. Anguelov. The waymo open sim agents challenge, 2023.

[5] S. Shi, L. Jiang, D. Dai, and B. Schiele. Motion transformer with global intention localization and local movement refinement. *Advances in Neural Information Processing Systems*, 2022.

[6] B. Varadarajan, A. Hefny, A. Srivastava, K. S. Refaat, N. Nayakanti, A. Cornman, K. Chen, B. Douillard, C. P. Lam, D. Anguelov, and B. Sapp. Multipath++: Efficient information fusion and trajectory aggregation for behavior prediction. In *2022 International Conference on Robotics and Automation (ICRA)*, pages 7814–7821, 2022.