Model Predictive Simulation Using Structured Graphical Models and Transformers

Xinghua Lou[1], **Meet Dave**[1], **Shrinu Kushagra**[1], **Miguel Lázaro-Gredilla**[1] **and Kevin Murphy**[1]
[1]Google DeepMind

We propose an approach to simulating trajectories of multiple interacting agents (road users) based on transformers and probabilistic graphical models (PGMs), and apply it to the Waymo SimAgents challenge. The transformer baseline is based on the MTR model (Shi et al., 2024), which predicts multiple future trajectories conditioned on the past trajectories and static road layout features. We then improve upon these generated trajectories using a PGM, which contains factors which encode prior knowledge, such as a preference for smooth trajectories, and avoidance of collisions with static obstacles and other moving agents. We perform (approximate) MAP inference in this PGM using the Gauss-Newton method. Finally we sample $K = 32$ trajectories for each of the $N \sim 100$ agents for the next $T = 8\Delta$ time steps, where $\Delta = 10$ is the sampling rate per second. Following the Model Predictive Control (MPC) paradigm, we only return the first element of our forecasted trajectories at each step, and then we replan, so that the simulation can constantly adapt to its changing environment. We therefore call our approach "Model Predictive Simulation" or MPS. We show that MPS improves upon the MTR baseline, especially in safety critical metrics such as collision rate. Furthermore, our approach is compatible with any underlying forecasting model, and does not require extra training, so we believe it is a valuable contribution to the community.

Keywords: Multi-Agent Planning, Probabilistic Graphical Model, Model Predictive Control, Transformer

Introduction

The use of transformers to create generative models to simulate agent trajectories, trained on large datasets such as Waymo Open Data (Ettinger et al., 2021), has become very popular in recent years. Most previous work has been focusing on improving the architecture (Nayakanti et al., 2023; Shi et al., 2024), the training objective (Ngiam et al., 2021; Shi et al., 2024), the trajectory representation (Philion et al., 2023; Seff et al., 2023) or the speed (Zhou et al., 2023) of these transformer-based models.

This paper tackles the problem from an orthogonal and complementary angle – namely the use of prior knowledge, encoded using a probabilistic graphical model (PGM). We perform approximate MAP inference in the PGM to "post process" the trajectory proposals from a base transformer model, to increase their realism and compliance with constraints, such as collision avoidance.

To ensure that our predicted forecasts are adaptive to the changing environment, we replan at each step, following the principle of model predictive control (MPC), which is widely used for controlling complex dynamical systems (Schwenzer et al., 2021). We therefore call our approach **Model Predictive Simulation (MPS)**.

Our MPS approach differs from previous PGM methods for trajectory simulation, such as JFP (Luo et al., 2023), in several ways. First, we explicitly include (data-dependent) factors for collision avoidance and smooth trajectories, so we have better control over the generated trajectories. Second, our approach is iterative (being based on MPC), while JFP commits to the trajectory proposals at $t = 0$ and is thus open loop. Third, our approach uses the Gauss-Newton method to compute the joint MAP estimate, whereas JFP is based on discrete belief propagation methods to choose amongst a finite set of candidate trajectories.

Method

Outer loop The overall simulation pseudocode is shown in Algo. 1. It generates a set of $K = 32$

Algorithm 1 SimAgents outer loop

Input Scene context $\mathbf{c}$, num. agents N, num. samples K, trajetory length T
Output Sampled trajectories, $h_{1:N}^{1:K,1:T}$

for $k = 1$ to K **do**
  $s_{1:N}^{k,0}$ = init-trajectory($\mathbf{c}$)
  for $t = 1$ to T **do**
    Sample $r_{1:N}^{k,t}$ = MPS($\mathbf{c}, s_{1:N}^{k,1:t-1}$)
    Extend $s_{1:N}^{k,1:t}$ = append($s_{1:N}^{k,1:t}, r_{1:N}^{k,t}$)
  end for
end for

trajectories, each of length $T = 80$, for N agents given the scene context $\mathbf{c}$. (The exact value of N depends on the number of agents that are visible in $\mathbf{c}$.) We denote the generated output by $s_{1:N}^{1:K,1:T}$, where $s_i^{k,t} = [x, y, \dot{x}, \dot{y}]$ is the state (2d location and velocity) of the i'th agent in sample k.

Algorithm 2 Model Predictive Simulation

Input Scene context $\mathbf{c}$, agent history $h_{1:N}$, num. agents N, future planning horizon F, number of rollouts J, transformer proposal π
Output Predicted next state for each agent, $r_{1:N}$

for $j = 1$ to J **do**
  Sample $(a_{1:N}^{j,1:F}, g_{1:N}^{j}) \sim \pi(\mathbf{c}, h_{1:N}, F)$
  G^j = BuildFactorGraph($a_{1:N}^{j,1:F}, g_{1:N}^{j}, \mathbf{c}$)
  Initialize $s_{1:N}^{j,1:F} = a_{1:N}^{j,1:F}$
  $(s_{1:N}^{j,1:F}, E^j)$ = Inference($G^j, s_{1:N}^{j,1:F}$)
end for
Sample $j^* \sim$ SoftMin($E^{1:J}$)
return $s_{1:N}^{j^*,1}$

Inner loop At each step t, the simulator calls our MPS algorithm to generate a prediction for the next state of each agent. The pseudocode for this is shown in Algo. 2. The approach is as follows. First we use the MTR transformer model π (Shi et al., 2024) to sample a set of N goal locations, $g_{1:N}^{j}$, one for each agent, as well as a sequence of anchor points leading to each goal, $a_{1:N}^{j,1:F}$, where F is the planning or forecast horizon. We do this $J = 60$ times in parallel, to create a set of possible futures. We then use the PGM to generate J joint trajectories (for all N agents), using the method described below. Finally we evaluate the energy of each generated trajectory, E^j, sample one of the low energy (high probability) ones to get $s_{1:N}^{j^*,1:F}$, and return the first step of this sampled trajectory, $s_{1:N}^{j^*,1}$.

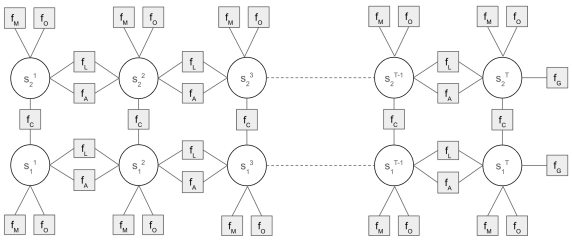

Figure 1 | Factor Graph for $N = 2$ agents unrolled for T planning steps. Circles are random variables, gray squares are fixed factors.

$$p(\mathbf{s}_{1..N}^{1:F} \mid \mathbf{c}, \mathbf{a}_{1:N}^{1:F}, g_{1:N}) \propto \prod_{i=1}^{N} \left[f_G(s_i^F | g_i) \cdot \prod_{t=1}^{F-1} f_M(s_i^t | a_i^t) \right] \cdot \prod_{i=1}^{N} \left[\prod_{t=2}^{F} f_L(s_i^{t-1}, s_i^t) \cdot f_A(s_i^{t-1}, s_i^t) \right] \cdot \prod_{i=1}^{N} \prod_{t=1}^{F} f_O(s_i^t \mid \mathbf{c}) \cdot \prod_{i \neq j} \prod_{t=1}^{F} f_C(s_i^t, s_j^t)$$

Figure 2 | Joint probability model.

Graphical model The key to our method is the probabilistic graphical model (PGM) for improving upon the proposed trajectories by MTR. The factor graph is shown in Fig. 1 and the corresponding conditional joint distribution is given in Fig. 2. The model was inspired by (Patwardhan et al., 2022) who uses Gaussian belief propagation. We now explain each of the factors.

First we have factors which compare a candidate trajectory to the original proposal. The motion factor is defined as $f_M(\mathbf{s}_i^t) = |s_i^t - a_i^t|$, where a_i^t is the predicted location (anchor point) for agent i at time t as computed by π. This ensures the trajectory stays close to the initial proposal. The proximity to goal factor is defined as $f_G(s_i^F) = |s_i^F - g_i|$, where g_i is the goal for agent i predicted by π. This ensures the trajectory ends close to where we expect.

Second we have factors defined from "physics". We define a factor that penalizes deviation from

linear motion: $f_{\mathrm{L}}(s, s') = |s'_{xy} - (s_{xy} + s_{\dot{x}\dot{y}}\Delta_t)|$, where s_{xy} are the location components of s, $s_{\dot{x}\dot{y}}$ are the velocity components of s, and Δt is the sampling rate. We also define a factor that penalizes change in direction: $f_{\mathrm{A}}(s, s') = |s_{\dot{x}\dot{y}} - s'_{\dot{x}\dot{y}}|$. We used weight 2.0 for f_{A} and 1.0 for all other factors.

Third we have factors derived from static obstacles on the road: $f_{\mathrm{O}}(s \mid \mathbf{c}) = \max_{(x,y)\in\mathbf{c}_{\mathrm{RE}}} \mathrm{G}(x, y \mid s)$, where $\mathbf{c}_{\mathrm{RE}}$ represents the coordinates of the road edges (part of the context $\mathbf{c}$) and $\mathrm{G}(x, y \mid s)$ is a Gaussian field centered and rotated according to the agent's location s_{xy}.

Finally, we have pairwise collision factors between agents: $f_{\mathrm{C}}(s, s') = \max_{(x,y)\in\mathrm{CCP}(s')} \mathrm{G}(x, y \mid s)$, where $\mathrm{G}(x, y \mid s)$ is a Gaussian field for agent s, and $\mathrm{CCP}(s')$ are the 9 collision checking points (CCP) for the other agent s' (4 corners, 4 centers of the sides, and center of the agent).

Inference Inference on the factor graph is equivalent to minimizing a non-linear, non-convex quadratic optimization problem defined over $\mathbf{s}_{1:N}^{1:F}$. For efficiency reasons, we developed a two-step approach. First, we use the Gauss–Newton method (Björck, 1996) to solve a partial model that only consists of f_{M}, f_{L} and f_{A} factors, as these can all be evaluated in parallel across agents using N individual trajectory models. This step produces smoothed trajectories, which are then frozen. Second, we sample joint trajectories for agents according to their probability (unnormalized energy), and use the f_{O} and f_{C} factors to score their quality. After repeating this J times, the best joint trajectories are sampled from a softmin operation over the scores of the J samples.

Experimental Evaluation

Benchmark We evaluated MPS on the 2024 Waymo Sim Agents Challenge, where the task is simulating 32 realistic rollouts of all agents in the scene given their 1s history for 8s into the future. The simulation needs to be closed-loop and factorized between the ADV and other agents, which MPS satisfies naturally.

Implementation Details We implemented the factors and the inference in JAX [1] and JAXopt [2] for the Gauss-Newton method. We leveraged JAX's just-in-time (JIT) compilation and observed great scalability. For speed up, we take 10 immediate next steps at each MPS iteration.

We trained our own MTR model π using the open source code [3] . We removed local attention and reduced the source polylines to 512. The training data is augmented by adding extra interacting agents, and by applying random history dropouts. We followed the original training setup except the number of epochs (50), the batch size (8) and the LR schedule ([25, 30, 35, 40, 45]). Training took about 3 days on 16 A100s.

We used only the official Waymo Open Motion Dataset v1.2.1 and did **not** use any Lidar or Camera data. We did **not** need any additional training and we did **not** use ensembles.

Sim Agents 2024 Results We ranked number 4 among all methods (Table 1). We outperformed the 2023 winner MVTE (Wang et al., 2023) which also uses MTR (Shi et al., 2024), and are approximately 1 point behind the 2024 winner SMART (Wu et al., 2024). MPS achieved near-top performance in a few safety critical metrics such as COLLISION and OFFROAD, showing the effectiveness of the priors in our model. MPS showed a lack of performance in LINEAR SPEED / ACCEL. We speculate this is because MPS can generate diverse rollouts that are very different from the logged data used for metric evaluation.

Ablation Study To evaluate the value of the PGM priors, we compare MPS to the same MTR model with random trajectory sampling (MTR+RAND) on the validation dataset. As shown in Table 2, MPS improved safety-critical metrics such as COLLISION, OFFROAD and the overall REALISM score, while lacked performance at LINEAR SPEED / ACCEL for the same reason discussed above.

Qualitative Study Qualitatively, MPS generates diverse (multi-modal) predictions (Fig. 3), and each prediction contains realistic traffic patterns such as lane merging, unprotected left turn, yielding, among others (Fig. 4).

[1] https://github.com/google/jax
[2] https://jaxopt.github.io/
[3] https://github.com/sshaoshuai/MTR